POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Deep learning based relative navigation about uncooperative space objects

Laurea Magistrale in Space Engineering - Ingegneria Spaziale

Author: Daniel Kaidanovic

Advisor: Prof. Pierluigi Di Lizia

Co-advisors: Michele Maestrini, Massimo Piazza

Academic year: 2020-2021

---

1. Introduction

As the number of space debris orbiting Earth is increasing, the awareness to this problem rose too in the latter years.
The space sector started to move towards a solution by planning more and more Active Debris Removal (ADR) and on-orbit servicing missions.
Since the approached bodies are tumbling objects with high angular velocities, a robust and accurate relative pose determination pipeline for uncooperative bodies is absolutely needed.

To this aim two main paths can be pursued:

- *Ground based determination*, which tracks the light or radar signature of the target to reconstruct its motion.
- *In-orbit determination*, which employs the on-board systems of a chaser spacecraft to determine the relative dynamics of the target body.

Unfortunately, due to higher uncertainties, the ground-based approach is not suited for close proximity operations. Thus leaving the in-orbit path as the only option.
To determine the relative pose and separation of a target body, within ranges closer than *50 m* a visual approach has to be pursued. Thus, either *Light Detection and Ranging (LiDAR)* devices, *Stereo cameras* or *Monocular cameras* have to be implemented.
LiDARs and Stereo cameras are generally bulky and expensive solutions, thus, the use of monocular cameras is suggested for simpler and cheaper missions. They offer a great alternative, although they require complex and computationally demanding image processing algorithms to properly work.
It is also important to highlight that the above-mentioned procedure needs to be robust and accurate to better deal with long approach sequences.

The aim of this thesis is to develop and test such pipeline starting from the Deep Learning based pipeline developed by Massimo Piazza in his MSc thesis [6].

2. Theoretical background

2.1. Relative distance problem

The relative distance problem describes how the separation between two spacecrafts orbiting a main body evolves over time in the *Local Verti-*

cal Local Horizontal (LVLH) frame.
In such reference frame the relative distance ($\mathbf{r_r}$) and the relative velocity ($\mathbf{v_r}$) can be expressed as:

$$\mathbf{r_r} = x\hat{\boldsymbol{r}} + y\hat{\boldsymbol{\theta}} + z\hat{\boldsymbol{h}} \tag{1}$$

$$\mathbf{v_r} = \dot{x}\hat{\boldsymbol{r}} + \dot{y}\hat{\boldsymbol{\theta}} + \dot{z}\hat{\boldsymbol{h}} \tag{2}$$

Where $[\hat{\boldsymbol{r}}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{h}}]$ are the versors of the considered LVLH triad.
The evolution of the relative distance between the two spacecrafts can then be expressed with the set of differential equations [7]:

$$\begin{cases} \ddot{x} = 2\dot{\nu}\dot{y} + \ddot{\nu}y + \dot{\nu}^2 x + \dfrac{\mu}{\bar{r}^2} \\ \qquad\qquad - \dfrac{\mu(\bar{r}+x)}{[(\bar{r}+x)^2 + y^2 + z^2]^{3/2}} \\ \ddot{y} = -2\dot{\nu}\dot{x} - \ddot{\nu}x + \dot{\nu}^2 y \\ \qquad\qquad - \dfrac{\mu y}{[(\bar{r}+x)^2 + y^2 + z^2]^{3/2}} \\ \ddot{z} = -\dfrac{\mu z}{[(\bar{r}+x)^2 + y^2 + z^2]^{3/2}} \end{cases} \tag{3}$$

Where μ is the standard gravitational parameter of the main attractor, ν is the true anomaly and $\bar{r}$ represents the distance between the main attractor and the chaser center.
To complete the problem, the orbital motion of the chaser must be described both in terms of true anomaly (ν) and position ($\bar{r}$) as:

$$\ddot{\bar{r}} = \bar{r}\dot{\nu}^2 - \frac{\mu}{\bar{r}^2} \tag{4}$$

$$\ddot{\nu} = -\frac{2\dot{\bar{r}}\dot{\nu}}{\bar{r}} \tag{5}$$

2.2. Relative attitude problem

The general equations governing the attitude motion of a rigid body are known as *Euler equations* and have the form:

$$\mathbf{J}\dot{\omega} + \omega \times \mathbf{J}\omega \ = \mathbf{M} \tag{6}$$

Where $\mathbf{J}$ represents the object inertia matrix, ω is its angular velocity, and $\mathbf{M}$ is the sum of the applied torques.
For the aim of this dissertation, *torque free dynamics* will be assumed by imposing $\mathbf{M} = \underline{0}$. This assumption is acceptable as disturbances are damped out by the attitude control of the chaser.
The notation relative to the angular velocities needs to be contextualized as [4]:

$$\begin{aligned} \omega_r &= \omega_t - \mathbf{\Gamma}\omega_c \\ \dot{\omega}_r &= \dot{\omega}_t - \mathbf{\Gamma}\dot{\omega}_c + \dot{\omega}_{app} \\ \dot{\omega}_{app} &= \omega_r \times \mathbf{\Gamma}\omega_c \end{aligned} \tag{7}$$

Where ω_t and ω_c are respectively the target and the chaser angular velocities expressed in their body fixed reference frame and ω_r represents the relative angular velocity expressed in the target body frame. Finally $\mathbf{\Gamma}$ is the rotation matrix linking the body-fixed reference frame of the target to the body-fixed reference frame of the chaser.
Furthermore, to describe the attitude this thesis makes use of the *Modified Rodrigues Parameters (MRPs)* notation, which is defined from the quaternions as:

$$\zeta = \frac{\tilde{\boldsymbol{q}}}{1+q_0} \quad \text{if } norm(\zeta) < 1 \tag{8}$$

$$\zeta_S = \frac{-\tilde{\boldsymbol{q}}}{1-q_0} \quad \text{if } norm(\zeta) > 1 \tag{9}$$

A relation connecting the MRPs and the aforementioned attitude matrix $\mathbf{\Gamma}$ is attained as:

$$\mathbf{\Gamma}(\zeta) = \boldsymbol{I}_3 - \alpha_1^A[\zeta\times] + \alpha_2^A[\zeta\times]^2 \tag{10}$$

where $\boldsymbol{I}_3$ is the Identity matrix and the parameters $\alpha_1^A, \alpha_2^A, [\zeta\times]$ can be obtained as:

$$\begin{cases} \alpha_1^A = 4\dfrac{1-\zeta^T\zeta}{(1+\zeta^T\zeta)^2} \\ \alpha_2^A = 8\dfrac{1}{(1+\zeta^T\zeta)^2} \\ [\zeta\times] = \begin{bmatrix} 0 & -\zeta_3 & \zeta_2 \\ \zeta_3 & 0 & -\zeta_1 \\ -\zeta_2 & \zeta_1 & 0 \end{bmatrix} \end{cases} \tag{11}$$

Finally the evolution of the relative attitude can be described in terms of MRPs and relative angular velocities as:

$$\begin{cases} \dot{\zeta}_r = \dfrac{1}{4}\Sigma(\zeta_r)\omega_r \\ \boldsymbol{J_t}\dot{\omega}_r + \omega_r \times \boldsymbol{J_t}\dot{\omega}_r = \\ \qquad\qquad \boldsymbol{M}_{app} - \boldsymbol{M}_g - \boldsymbol{M}_{ci} \end{cases} \tag{12}$$

where $\boldsymbol{J_t}$ is the inertia matrix of the target, ζ_r

is the MRP describing the relative attitude and:

$$\begin{aligned}\Sigma(\zeta) &= (1-\zeta^T\zeta)\boldsymbol{I}_3 + 2\zeta\zeta^T + 2[\zeta\times] \\ \boldsymbol{M}_{app} &= \boldsymbol{J_t}\dot{\omega}_r \times \Gamma\omega_c \\ \boldsymbol{M}_g &= \Gamma\omega_c \times \boldsymbol{J_t}\Gamma\omega_c + \\ &\quad (\omega_r \times \boldsymbol{J_t}\Gamma\omega_c + \Gamma\omega_c \times \boldsymbol{J_t}\omega_r) \\ \boldsymbol{M}_{ci} &= \boldsymbol{J_t}\Gamma\dot{\omega}_c\end{aligned} \tag{13}$$

where $\boldsymbol{M}_{app}$ represents the apparent torque, $\boldsymbol{M}_g$ are the gyroscopic torque and $\boldsymbol{M}_{ci}$ are the chaser-inertial torque.

2.3. Kalman filtering

The Kalman filter and its variants are within the most used and efficient recursive filters in the space sector. In its basic form the Kalman Filter (KF) [3] is defined as a *predictor-corrector* algorithm as it is composed of both a prediction and a correction phase.
Starting from a linear discrete-time system in the form:

$$\begin{cases}\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{u}_k + \mathbf{w}_{(k)} \\ \mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k\end{cases} \tag{14}$$

Where $\mathbf{x}_k$ and $\mathbf{u}_k$ represent respectively the state and the input at timestep k, instead, $\mathbf{w}_k$ and $\mathbf{v}_k$ are used to identify the process noise and the measurement noise, whereas $\mathbf{y}_k$ identifies the measurement output at timestep k. Finally the matrices $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{H}$ respectively represent the State transition matrix, the Input transition matrix and the Observation matrix.
The aforementioned *Prediction phase* performs a propagation of the state and the covariance ($\hat{\mathbf{x}}_k^+$, $\mathbf{P}_k^+$) based on their values in the previous iteration as:

$$\hat{\mathbf{x}}_{k+1}^- = \mathbf{F}\hat{\mathbf{x}}_k^+ + \mathbf{G}\mathbf{u}_k \tag{15}$$

$$\mathbf{P}_{k+1}^- = \mathbf{F}\mathbf{P}_k^+\mathbf{F}^T + \mathbf{Q}_k \tag{16}$$

Which returns the expected mean value of the state $\hat{\mathbf{x}}_{k+1}^-$ and the covariance matrix $\mathbf{P}_{k+1}^-$ for the current iteration.
Note that $\mathbf{Q}_k$ represents the covariance matrix associated to the process noise, while $\mathbf{R}_k$ is associated to the measurement noise.
The next step is defined as *Correction step* as it corrects the predicted state and covariance matrix. It does so by exploiting the incoming measurements and a weighting parameter defined as *Kalman gain* ($\mathbf{K}_{k+1}$).
The steps to be followed are:

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1}^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_{k+1}^-\mathbf{H}^T + \mathbf{R}_{k+1})^{-1} \tag{17}$$

$$\hat{\mathbf{x}}_{k+1}^+ = \hat{\mathbf{x}}_{k+1}^- + \mathbf{K}_{k+1}(\mathbf{y}_{k+1} - \mathbf{H}\hat{\mathbf{x}}_{k+1}^-) \tag{18}$$

$$\mathbf{P}_{k+1}^+ = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H})\mathbf{P}_{k+1}^- \tag{19}$$

Note that the output of the *Correction step* becomes the input of the next filter iteration as the filter uses recursive information to improve its guesses.

2.3.1 Extended Kalman Filter

As the simple KF cannot deal with non-linear problems some alternative architectures had to be devised.
A common solution to this issue is the *Extended Kalman Filter (EKF)* [5] which adopts a linearization of the problem to get around the present non-linearities.
Starting from the non-linear model:

$$\begin{cases}\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \\ \mathbf{y}_k = h(\mathbf{x}_k, \mathbf{v}_k)\end{cases} \tag{20}$$

The Jacobian matrices of the state transition function $f(\cdot)$ and the measurement function $h(\cdot)$ can be computed as:

$$\mathbf{F} = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^+} \qquad \mathbf{H} = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^-} \tag{21}$$

Once the Jacobians have been computed the process becomes very similar to the one adopted by the KF.
The performed steps are:

$$\hat{\mathbf{x}}_{k+1}^- = f(\hat{\mathbf{x}}_k^+, \mathbf{u}_k, 0) \tag{22}$$

$$\mathbf{P}_{k+1}^- = \mathbf{F}\mathbf{P}_k^+\mathbf{F}^T + \mathbf{Q}_k \tag{23}$$

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1}^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_{k+1}^-\mathbf{H}^T + \mathbf{R}_{k+1})^{-1} \tag{24}$$

$$\hat{\mathbf{x}}_{k+1}^+ = \hat{\mathbf{x}}_{k+1}^- + \mathbf{K}_{k+1}(\mathbf{y}_{k+1} - h(\hat{\mathbf{x}}_{k+1}^-, 0)) \tag{25}$$

$$\mathbf{P}_{k+1}^+ = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H})\mathbf{P}_{k+1}^- \tag{26}$$

2.3.2 Unscented Kalman Filter

For particularly non-linear systems the EKF may fall short, thus a new filtering architecture must be devised.
By leveraging the *Unscented Transform (UT)*

for the propagation of the state and covariance, the *Unscented Kalman Filter (UKF)* [2] can be defined.
Considering a generic nonlinear system:

$$\begin{cases} \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \\ \mathbf{y}_k = h(\mathbf{x}_k, \mathbf{v}_k) \end{cases} \tag{27}$$

according to the UT a set of sigma points around the state must be defined as:

$$\hat{\mathbf{x}}_k^{(i)} = \hat{\mathbf{x}}_k^+ + \tilde{\mathbf{x}}^{(i)}, \qquad i = 1, ..., 2n \tag{28}$$

$$\tilde{\mathbf{x}}^{(i)} = \left(\sqrt{c\mathbf{P}_k^+}\right)_i^T, \qquad i = 1, ..., n \tag{29}$$

$$\tilde{\mathbf{x}}^{(n+i)} = -\left(\sqrt{c\mathbf{P}_k^+}\right)_i^T, \qquad i = 1, ..., n \tag{30}$$

where c is a tuning parameter used to define the spread and weight of each sigma point.
The latter are then propagated and a weighted average is computed to define the predicted state and covariance as:

$$\hat{\mathbf{x}}_{(k+1)}^{(i)} = f(\hat{\mathbf{x}}_k^{(i)}, \mathbf{u}_{k+1}) \tag{31}$$

$$\hat{\mathbf{x}}_{k+1}^- = \sum_{i=0}^{2n} W_M^{(i)} \hat{\mathbf{x}}_{k+1}^{(i)} \tag{32}$$

$$\mathbf{P}_{k+1}^- = \sum_{i=0}^{2n} W_c^{(i)} (\hat{\mathbf{x}}_{k+1}^{(i)} - \hat{\mathbf{x}}_{k+1}^-) (\hat{\mathbf{x}}_{k+1}^{(i)} - \hat{\mathbf{x}}_{k+1}^-)^T + \mathbf{Q}_k \tag{33}$$

where $W_M^{(i)}$ and $W_c^{(i)}$ are weighting parameters.
The obtained predicted state and covariance are then used to correct the sigma points as:

$$\hat{\mathbf{x}}_{k+1}^{(i)} = \hat{\mathbf{x}}_{k+1}^- + \tilde{\mathbf{x}}^{(i)}, \qquad i = 1, ..., 2n \tag{34}$$

$$\tilde{\mathbf{x}}^{(i)} = \left(\sqrt{c\mathbf{P}_{k+1}^-}\right)_i^T, \qquad i = 1, ..., n \tag{35}$$

$$\tilde{\mathbf{x}}^{(n+i)} = -\left(\sqrt{c\mathbf{P}_{k+1}^-}\right)_i^T, \qquad i = 1, ..., n \tag{36}$$

The new set of sigma points is then used in the correction step to define: the expected measurement, the measurement covariance $\mathbf{P}_y$ and the cross covariance $\mathbf{P}_{xy}$.
This is done as:

$$\hat{\mathbf{y}}_{k+1}^{(i)} = h(\hat{\mathbf{x}}_{k+1}^{(i)}) \tag{37}$$

$$\hat{\mathbf{y}}_{(k+1)}^- = \sum_{i=1}^{2n} W_M^{(i)} \hat{\mathbf{y}}_{k+1}^{(i)} \tag{38}$$

$$\mathbf{P}_y = \sum_{i=1}^{2n} W_c^{(i)} (\hat{\mathbf{y}}_{k+1}^{(i)} - \hat{\mathbf{y}}_{k+1}^-) (\hat{\mathbf{y}}_{k+1}^{(i)} - \hat{\mathbf{y}}_{k+1}^-)^T + \mathbf{R}_k \tag{39}$$

$$\mathbf{P}_{xy} = \sum_{i=1}^{2n} W_c^{(i)} (\hat{\mathbf{x}}_{k+1}^{(i)} - \hat{\mathbf{x}}_{k+1}^-) (\hat{\mathbf{y}}_{k+1}^{(i)} - \hat{\mathbf{y}}_{k+1}^-)^T \tag{40}$$

These quantities are then used to define a Kalman gain and correct the state and covariance as:

$$\mathbf{K}_{k+1} = \mathbf{P}_{xy} \mathbf{P}_y^{-1} \tag{41}$$

$$\hat{\mathbf{x}}_{k+1}^+ = \hat{\mathbf{x}}_{k+1}^- + \mathbf{K}_{k+1}(\mathbf{y}_{k+1} - \hat{\mathbf{y}}_{k+1}^-) \tag{42}$$

$$\mathbf{P}_{k+1}^+ = \mathbf{P}_{k+1}^- - \mathbf{K}_{k+1} \mathbf{P}_y \mathbf{K}_{k+1}^T \tag{43}$$

3. Relative Pose pipeline

The work proposed in this thesis aims to create a supporting structure for the Deep learning pipeline proposed by Piazza [6].
To do so an image generation pipeline to provide the inputs and a filtering procedure to refine the outputs have been created.

Starting from the image generation, a *Blender* based architecture has been proposed.
The pipeline has the aim of generating accurate spaceborne images of the TANGO spacecraft at different relative distances and orientations.

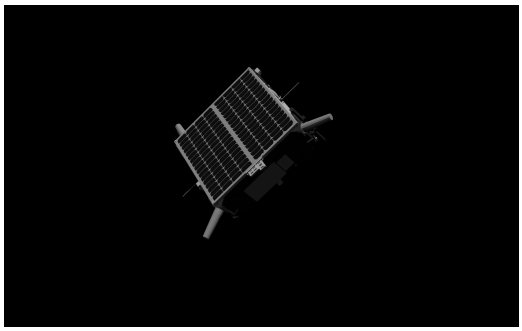


Figure 1: Example of a generated frame

Furthermore, the algorithm has been provided with the capability of generating single uncorrelated frames or full coherent sequences, as both modes were needed for testing and validation purposes.
The pipeline, shown in Image 3, starts by determining the relative position of the artificial camera with respect to the target body frame.

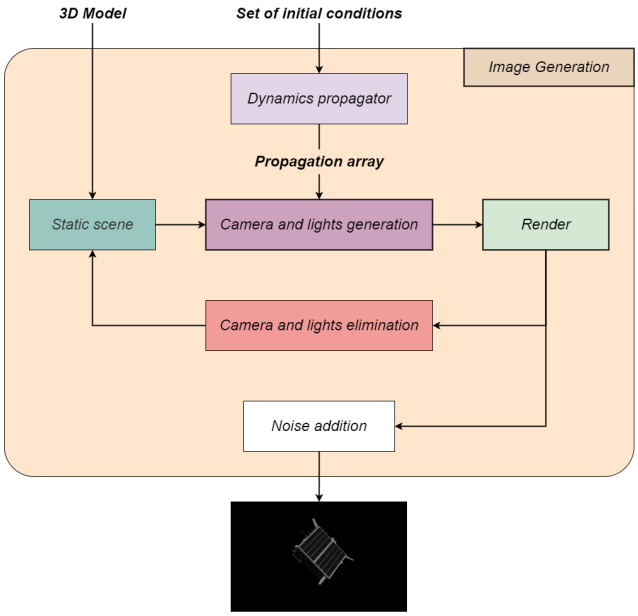


Figure 2: Scheme of the image generation pipeline

Subsequently it generates it in the scene and orients it based on the attitude matrix provided. Afterwards, considering the orbital position of the chaser and the considered season, a set of custom lights is placed in the scene to mimic realistic lighting conditions.
Once all components are put in place a rendered frame is generated using the *EEVEE* engine. Afterwards a noise filter is applied, then the image is greyscaled and saved.
Finally, both the camera and the set of lights are removed from the scene and a new rendering procedure can start.
The position and attitude of the camera can be provided for both a static database or a coherent sequence. This is done by following two different procedures:

- *Static database*, the attitude and position data is provided from a pre-generated json database and the pipeline treats each frame individually
- *Image sequence*, the initial attitude and position are provided as well as a sequence length and its correlated *frames per second (fps)* rate.
  The sequence is then propagated in time following Equations 3 and 12.
  Finally, measurements are extracted and loaded onto the pipeline according to the imposed fps rate.

Moving on to the filtering pipeline, it consists in a KF based architecture whose aim is to improve the output coming from the *Neural Network (NN)* pipeline.

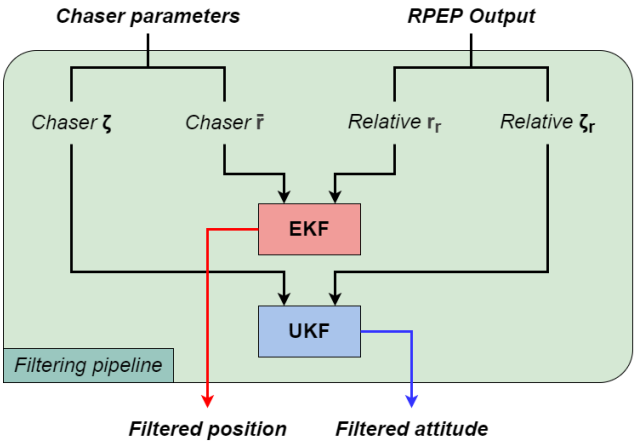


Figure 3: Scheme of the filtering pipeline

As can be seen from Figure 3 it was chosen to separate and apply different filtering techniques to the relative distance and the relative attitude problem. This was enabled by assuming that the attitude of the chaser is perfectly known and available.
This approach was pursued as the relative pose presented a stronger non-linear behaviour with respect to the simpler relative distance problem, meaning that different filtering techniques had to be applied. In detail, the EKF was applied to the simpler and more linear relative separation problem, whereas the UKF was deemed fit to deal with the more complex relative pose problem.

4. Results

The first step of the testing campaign was to make sure the image generation pipeline was up to par.
To this aim the NN pipeline was employed to compare the results obtained from the *SPEED* database [1] and an analogous one generated by the image generation pipeline.
The generation of such database was possible as the relative parameters associated to each frame of the *SPEED* database were available.
Some of the results obtained from such comparison are reported in Tables 1 and 2.

Absolute error	
	Mean
E_t	*10.36 cm*
$\boldsymbol{E_t}$	*[0.52, 0.56, 10.25] cm*
E_θ	*2.24°*
$\boldsymbol{E_\theta}$	*[1.57°, 0.84°, 1.72°]*
Standard Deviation	
$\boldsymbol{\sigma_{E_t}}$	*[1.62, 1.71, 30.44] cm*
$\boldsymbol{\sigma_{E_\theta}}$	*[8.92°, 5.11°, 10.82°]*

Table 1: Results obtained from the *SPEED* database

Absolute error	
	Mean
E_t	*10.10 cm*
$\boldsymbol{E_t}$	*[0.58, 0.65, 9.96] cm*
E_θ	*2.32°*
$\boldsymbol{E_\theta}$	*[1.65°, 0.83°, 1.65°]*
Standard Deviation	
$\boldsymbol{\sigma_{E_t}}$	*[1.79, 2.45, 39.82] cm*
$\boldsymbol{\sigma_{E_\theta}}$	*[9.41°, 4.72°, 9.91°]*

Table 2: Results obtained from the simulated database

Overall the results turned out similar with a minor loss of performance on the simulated database, thus the image generation pipeline was successfully validated.
Following this, a set of dynamic sequences were generated and evaluated by the NN pipeline.
This process was performed in order to provide further validation, and to produce a set of real input baselines to test the filters.
The obtained results are reported in Table 3.

Absolute error	
	Mean
E_t	*8.36 cm*
$\boldsymbol{E_t}$	*[0.50, 0.48, 8.26] cm*
E_θ	*2.74°*
$\boldsymbol{E_\theta}$	*[1.47°, 0.69°, 2.47°]*
Standard Deviation	
$\boldsymbol{\sigma_{E_t}}$	*[0.70, 1.22, 15.69] cm*
$\boldsymbol{\sigma_{E_\theta}}$	*[5.22°, 1.30°, 14.44°]*

Table 3: Results obtained from a sample sequence

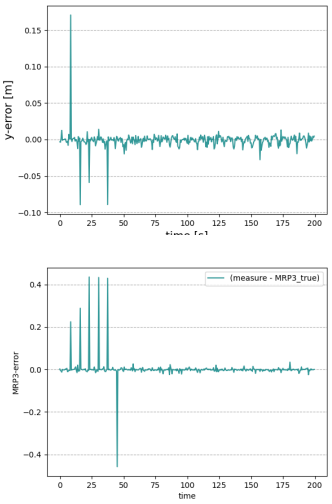


Figure 4: Graphs of the distance [*top*] and attitude [*bottom*] error

As can be seen from the Table 3 the NN results were in line with the expected ones. However, as highlighted in Figure 4, some outliers were also detected.
The obtained results were then filtered to observe the quality of the filtering pipeline.
The filtering action visualized in Figure 5 has been obtained considering: an acquisition rate of *2 fps*, a rotational period of *10 s* and initial conditions taken from *img000040* of the *SPEED* database.

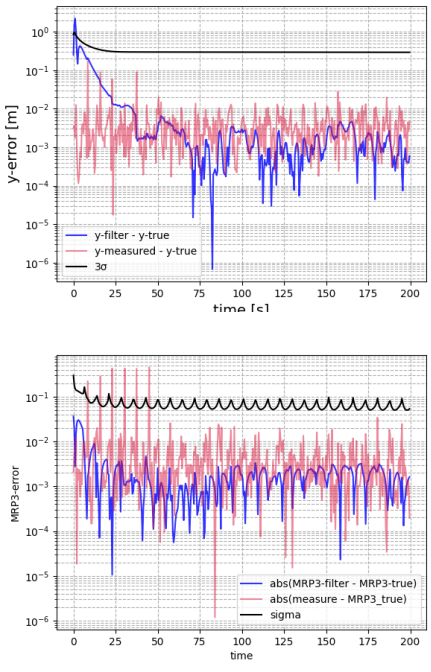


Figure 5: Graphs of the EKF [*top*] and UKF [*bottom*] filtering performance

As can be seen from Figure 5 both the EKF and the UKF managed to successfully filter out any outlier and improve the accuracy of the results of up to one order of magnitude.
Despite the promising results, it was chosen to further evaluate the robustness and reliability of the filtering pipeline.
This was accomplished by performing thousands of runs on pseudo-measurements, which were generated after determining the *error covariance matrix (R)* associated to the NN pipeline.
Overall the EKF proved to be extremely reliable and robust, never failing during thousands of tests. This performance was attributed to the slower and simpler dynamics it had to deal with.
Instead the UKF performance was tested over an array of different *rotational periods* (t_{rot}) of the target spacecraft. Overall its performance was deemed acceptable, however a number of failures were detected, especially for faster t_{rot}.
The number of detected failures is reported in Table 4.

UKF testing results			
t_{rot}	***Tests***	***Failures***	***Fail rate***
15 s	*1000*	*24*	*2.4 %*
10 s	*1000*	*39*	*3.9 %*
5 s	*1000*	*83*	*8.3 %*

Table 4: Results from the UKF testing campaign

To better understand the underlying mechanisms triggering the observed failures, a dedicated analysis was performed. Overall four main failure modes were detected and studied:

- *Localized spike*, [≈ 65%], consists in an isolated sudden increase of the error followed by its instantaneous falloff (see Figure 6).

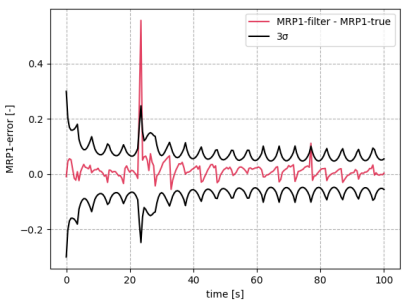


Figure 6: Example of a *Localized spike*

- *Extended failure*, [≈ 30%], consists in a prolonged failure caused by a sequence of successive outliers (see Figure 7).

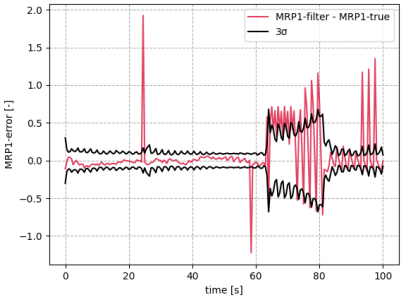


Figure 7: Example of an *Extended failure*

- *Initial failure*, a particular case of *Extended failure* where it is detected in the very first iterations of the UKF (see Figure 8).

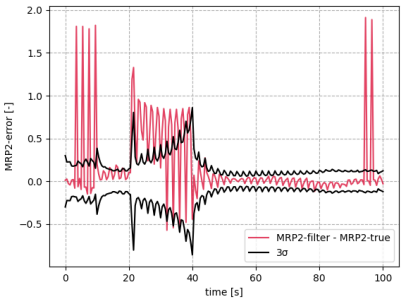


Figure 8: Example of an *Initial failure*

- *Total fail*, [≈ 5%], consists in the complete divergence of the filter. It is generally caused by an extended sequence of outliers or a very bad initial guess (see Figure 9).

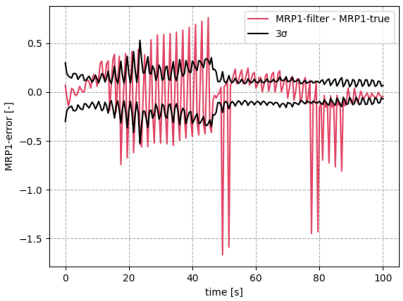


Figure 9: Example of a *Total fail*

Finally, an attempt at running the pipeline has been carried out on hardware that is closer in performance to space-grade components.

The chosen platform was the *Raspberry Pi 4 2GB* single board computer, running the *Raspberry Pi OS 64bit* operating system.
Both the NN pipeline and the filtering pipeline were tested and timed.
The obtained results can be found in Table 5.

Raspberry Pi performance			
NN pipeline	**EKF**	**UKF**	**Total**
92.71 s	*0.02 s*	*0.37 s*	*93.10 s*

Table 5: *Raspberry Pi 4 2GB* performance

Overall the performance is very poor, but it is important to highlight that the whole process was completely un-optimized.

An improvement in the performance of up to a factor *100* is to be expected once proper optimization procedures are adopted. Those could range from the low-level C/C++ implementation of the algorithm to the use of dedicated analog matrix processors to speed up the NN phase.

5. Conclusion

This thesis tackled the problem of relative attitude determination of an uncooperative spacecraft by expanding a pre-existing NN pipeline.
A way to generate realistic spaceborne images was implemented and tested extensively.
Moreover a filtering pipeline based on a set of UKFs and EKFs was implemented in order to improve the performance of the whole system.
Those were also extensively tested on both real and simulated inputs in order to isolate and analyze any observed failure mode.
In the end the whole pipeline was uploaded to a *Raspberry Pi 4 2GB* single board computer to evaluate its performance on more significant space hardware.

6. Acknowledgements

First of all I would like to express my gratitude to my advisor professor Pierluigi Di Lizia for his constant support and passion for the work I carried out.
Furthermore I would like to extend my gratitude to my co-advisors Michele Maestrini and Massimo Piazza for their absolute availability, precious insights and constant help.
Finally, I would like to thank OHB Sweden for the permission to use the 3D model of the TANGO spacecraft throughout the duration of this thesis.

References

[1] ACT and SLAB. Pose estimation challenge 2019. `https://kelvins.esa.int/satellite-pose-estimation-challenge/home/`, 2019.

[2] S.J. Julier, J.K. Uhlmann, and H.F. Durrant-Whyte. A new approach for filtering nonlinear systems. 3, 1995.

[3] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960.

[4] Mauro Massari and Mattia Zamarro. Application of sdre technique to orbital and attitude control of spacecraft formation flying. *Acta Astronautica*, 94(1):409–420, 2014.

[5] O. Montenbruck and E. Gill. Satellite orbits: Models, methods, and applications. *Applied Mechanics Reviews*, 55(2), 04 2002.

[6] Massimo Piazza. Deep learning-based monocular relative pose estimation of uncooperative spacecraft. Master's thesis, Politecnico di Milano, 11 2020.

[7] Hanspeter Schaub and John L Junkins. *Analytical mechanics of space systems*. Aiaa, 2003.

POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

AI-aided optical navigation about uncooperative spacecraft using synthetic imagery

Tesi di Laurea Magistrale in
Space Engineering - Ingegneria Spaziale

Author: **Daniel Kaidanovic**

Student ID: 945046
Advisor: Prof. Pierluigi Di Lizia
Co-advisors: Michele Maestrini, Massimo Piazza
Academic Year: 2020-21

Abstract

Nowadays the space sector is experiencing an increase in the demand for missions involving proximity operations. This trend can be partially attributed to the raised awareness of the space debris problem, as more and more Active Debris Removal (ADR) and on-orbit servicing missions are being planned. Consequently, the need for an accurate onboard relative navigation system is increasingly present in the industry.
This work proposes a pipeline for relative navigation based on Deep Learning techniques to obtain the relative pose measurements and Kalman Filtering to reconstruct the relative dynamics and add robustness to the pipeline. Furthermore, a testing procedure involving a Blender-based spaceborne image generator has been devised and applied to validate the results in the case of a realistic image sequence.

The overall pipeline is based on a pre-existing Neural Network pipeline that participated in ESA Pose Estimation Challenge with excellent results. This network achieved a centimeter-level position accuracy and degree-level attitude accuracy, along with considerable robustness to changes in background and lighting conditions.

To reconstruct the state during navigation, a set of Kalman filters have been implemented to tackle attitude and position separately.
For the relative distance, an Extended Kalman Filter has been applied, as the underlying relative dynamics can be accurately modeled by means of a linearized model. Instead, for the more complicated attitude problem, the choice fell on the Unscented Kalman Filter thanks to its superior robustness in highly non-linear dynamics.
In addition, robustness was taken as a priority with thousands of tests aimed at identifying and counteracting the most common failure modes. Moreover, some techniques were also developed for the detection and rejection of measurement outliers.

The whole navigation pipeline was then tested on a simulated set of image sequences of the TANGO spacecraft in torque-free tumbling conditions. The frames were obtained from a Blender-based spaceborne image generation platform exploiting a 3D model of the target and relying on an accurate propagation of the relative dynamics.
Finally, this work also presents the preliminary results coming from the implementation

of the pipeline on a Raspberry Pi 4 single-board computer for a preliminary evaluation of its performance on more representative hardware. The results, although not directly applicable for real-time navigation, proved to be promising.

Overall the pipeline managed to leverage the predictions of the Neural Network it was based on, adding robustness and precision with a minor addition of computational time.

Keywords: Neural Networks, spaceborne image generation, relative attitude determination, Kalman filtering.

Abstract in lingua italiana

Al giorno d'oggi il settore spaziale sta registrando un aumento nella domanda di missioni che coinvolgono operazioni di prossimità. Questa tendenza può essere in parte attribuita alla maggiore consapevolezza del problema dei detriti spaziali, poiché sempre più missioni di recupero attivo e missioni di manutenzione in orbita sono programmate. Di conseguenza, la necessità di un accurato sistema di navigazione relativa di bordo è sempre più presente nel settore.
Questo lavoro propone un algoritmo per la navigazione relativa basata su tecniche di Deep Learning per ottenere le misurazioni di posa relativa e filtri di Kalman per ricostruire la dinamica relativa e aggiungere robustezza. Inoltre, è stata ideata e applicata una procedura di test che coinvolge un generatore di immagini spaziali basato su Blender per convalidare i risultati nel caso di una sequenza realistica di immagini.

Nel complesso l'algoritmo si basa su una rete neurale preesistente che ha partecipato alla "Pose Estimation Challenge" dell'ESA con ottimi risultati. Questa rete ha raggiunto una precisione centimetrica sulla posizione e una precisione di assetto a livello di gradi, insieme a una notevole robustezza ai cambiamenti delle condizioni di sfondo e di illuminazione.

Per ricostruire lo stato durante la navigazione, è stata implementata una serie di filtri di Kalman per affrontare separatamente i problemi di assetto relativo e posizione relativa.
Per la distanza relativa è stato applicato un filtro di Kalman esteso EKF, in quanto le dinamiche relative sottostanti sono rappresentate da un modello dinamico linearizzato.
Per il problema di assetto, più complesso del precedente, la scelta è caduta sull'Unscented Kalman Filter (UKF), grazie alla sua superiore robustezza nel trattare dinamiche altamente non lineari.
Inoltre, l'affidabilità dell'algoritmo è stata considerata una priorità, difatti migliaia di test volti a identificare e contrastare le modalità di fallimento più comuni sono stati effettuati. In aggiunta, sono state sviluppate anche alcune tecniche per il rilevamento e lo scarto di valori anomali nelle misure. L'intero algoritmo di navigazione è stato quindi testato su un insieme simulato di sequenze di immagini del satellite TANGO in condizioni di moto incontrollato. I fotogrammi sono stati ottenuti da una piattaforma di generazione di immagini spaziali basata su Blender sfruttando un modello 3D del target e basandosi

su un'accurata propagazione delle relative dinamiche.

Infine, questo lavoro presenta anche i risultati preliminari provenienti dall'implementazione della pipeline su un computer "single-board" Raspberry Pi 4 per una valutazione preliminare delle sue prestazioni su hardware più rappresentativi. I risultati si sono rivelati promettenti, sebbene non direttamente applicabili per la navigazione in tempo reale.

Nel complesso, l'algoritmo è riuscito a sfruttare le previsioni della rete neurale su cui si basava, aggiungendo robustezza e precisione con un leggero incremento nel carico computazionale.

Parole chiave: Neural Networks, generazione di immagini spaziali, determinazione della posa relativa, filtri di Kalman.

Acknowledgements

First of all I would like to express my gratitude to my advisor, Professor Pierluigi Di Lizia, for his constant support and passion for the work I carried out.
Furthermore, I would like to extend my gratitude to my co-advisors Michele Maestrini and Massimo Piazza for their absolute availability, precious insights and constant help.
I would like to thank OHB Sweden for the permission to use the 3D model of the TANGO spacecraft throughout the duration of this thesis.
I would also like to show appreciation to my Polimi mates, Cecilia G., Eleonora G., Ludovica L., Alessio N., Marta P. for always being with me throughout this long journey.
A special thanks goes to the group of the *never ended sessions*: Alessio G., Lorenzo G., Michele G., Pietro G., Tommaso L., with whom I hope to share again a lot of fun and light-hearted moments.
I want to extend my gratitude to all the close friends in my hometown, Jermaine L., Beatrice C., Davide C., Claudia D., Beatrice N., Francesco N., thank you for bringing color to the grayness of Lodi.
An honorable mention goes to the *Voidless* team, Mattia B., Guglielmo R., Carlo V. and to all our mentors. I am sure that we will soon live in an overpackaging free world.
A very special thank you goes to Chiara for her constant, unconditional support and for always pointing out the brighter side of things. I feel very blessed with you by my side.
Last but not least i want to thank my family. Thank you Anna and Stefano for always being there through thick and thin. I would not be who I am now without your guidance. Thank you Eilin for always being my sidekick. Thank you Luisa, Piero, Генрик and Лидия for always encouraging me to follow my path. Thank you Mirek for the big contribution to this milestone. Thanks to Michela, Gabriele, Anna and the rest of my family for believing so much in me.

Contents

Acronyms

ADR Active Debris Removal. i, 1, 2

ANN Artificial Neural Network. 19, 21

BB Bounding Box. 21, 22, 23, 24, 25, 38, 48, 60

CNN Convolutional Neural Network. vii, xiii, 7, 18, 19, 20, 21, 22, 23, 69, 71

DCM Direction Cosine Matrix. 11

DoF Degrees of Freedom. 4

ECI Earth Centered Inertial. 37

EKF Extended Kalman Filter. iii, xiv, 9, 26, 27, 28, 29, 43, 44, 45, 54, 55, 56, 57, 59, 61, 69, 72, 83

EPnP Efficient Perspective-n-Point. vii, 7, 21, 24, 68

ESA European Space Agency. 4, 5, 25

FPGA Field Programmable Gate Arrays. 69

fps Frames Per Second. 41, 42, 62, 69, 71

GT Ground Truth. 19, 23, 54, 56, 57

HRNet High Resolution Network. xiii, 21, 23, 24

IoU Intersection over Union. 23

ITM Input transition matrix. 26

JUICE JUpiter ICy moons Explorer. 5

KF Kalman Filter. 8, 26, 27, 28, 71

LEO Low Earth Orbit. 40, 61

LiDAR Light Detection And Ranging. 2

LRN Landmark Regression Network. vii, 18, 21, 23, 24, 60, 68

LVLH Local Vertical Local Horizontal. 14, 40

MEKF Multiplicative Extended Kalman Filter. 12

MRP Modified Rodrigues Parameter. vii, xiii, xiv, 12, 13, 14, 16, 17, 45, 46, 51, 53, 58, 59, 60, 61, 63, 64, 65, 66, 67

NN Neural Networks. vii, xiii, 8, 18, 19, 33, 69

PANGU Planet and Asteroid Natural Scene Generation Utility. 4

PRISMA Prototype Research Instruments and Space Mission technology Advancement. 5, 7

RNN Recursive Neural Network. 19

RPEP Relative Pose Estimation Pipeline. viii, xiii, xv, 18, 21, 22, 24, 25, 40, 44, 47, 48, 49, 50, 51, 52, 54, 56, 57, 58, 59, 60, 68, 69, 71, 72

SLAB Space Rendezvous Laboratory. 4, 5, 25

SLN Spacecraft Localization Network. vii, xiii, 18, 21, 22, 23, 24, 37, 60, 68

SPEED Spacecraft PosE Estimation Dataset. viii, xiii, xiv, xv, 5, 6, 22, 23, 25, 33, 38, 39, 40, 41, 47, 50, 52, 56, 71

STM State Transition Matrix. 26

TRON Testbed for Rendezvous and Optical Navigation. 4, 5, 6

UKF Unscented Kalman Filter. iii, xiv, xv, 9, 26, 28, 29, 31, 43, 44, 45, 57, 58, 59, 61, 62, 63, 64, 66, 68, 69, 72, 83

UT Unscented Transform. xiii, 9, 28, 29

YOLO You Only Look Once. 21, 22, 24

List of Figures

List of Tables

1 | Introduction

1.1 Problem statement

Nowadays the space sector is experiencing an increase in the demand of missions involving close proximity operations. This trend can be partially attributed to the raised awareness of the space debris problem, as more and more Active Debris Removal (ADR) and on-orbit servicing missions are being planned [ML21].

On-orbit satellite servicing refers to a mission that has the aim of refueling, repairing and generally extend the lifetime of in orbit satellites.

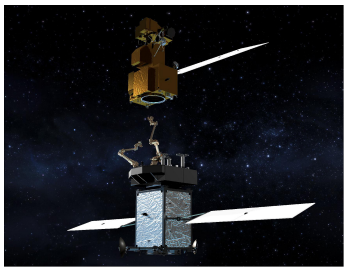


Figure 1.1: Rendering of the OSAM-1 servicer [bottom] (image credit: NASA)

As is often the case, the target satellite is an active one and thus is capable of establishing a communication link with the servicing spacecraft (defined as Chaser). This allows an exchange of information on both status and attitude that allows a strong cooperation between the two satellites and generally less risks.
Modern examples of this missions are Northrop Grumman's *Mission Extension Vehicle-1 and -2* [Gru21], which are both successfully carrying out their missions as of now and also NASA's *OSAM-1 mission (ex.Restore-L)* [Ree+16] whose launch is planned in 2025.

ADR refers to a mission that has the aim of capturing a space debris in order to deorbit it and effectively dispose of it. The target is considered uncooperative, as no communication can be established and no facilitating measures (e.g. light beacons) can be enacted by the target.
Moreover, it is quite often a tumbling body with unknown attitude, dynamics, and high angular velocities, making the mission very difficult and quite risky.

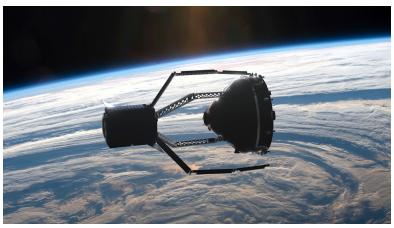


Figure 1.2: Rendering of the Clearspace mission (image credit: [ESA19a])

One of the most famous examples is ESA's *ClearSpace mission* [ESA19a] scheduled for 2026 which will be the first ADR mission ever.
To ensure the feasibility of these types of missions, a relative pose estimation pipeline, that does not rely on the collaboration of the target spacecraft, must be implemented and must ensure high performance and accuracy.

To this aim two main methods can be used. Firstly a ground based estimation can be considered: it consists in tracking the light signature of a target through the use of ground-based radar facilities and then reconstructing its motion from the gathered data.
Unfortunately, this method is highly dependent on the target visibility, is associated with very high uncertainties and the actual pose determination is virtually impossible. These drawbacks make this approach unsuited to be applied during the close proximity phase of a space mission.
The only alternative is basing the pose estimation on the chaser sensors, generally *Cameras* and *LiDAR* are the best suited for this tasks [Opr+17].
Light Detection And Ranging (LiDAR) sensors, both scanning and flashing, have been studied for years and implemented with regards to relative pose estimation. They are: very robust to different lighting conditions, can work at very high frame rates and generate a cloud of points which is fitted to a known 3D model of the target to extrapolate the relative pose [ZXB18].
Although they are very capable and promising, they are quite heavy, very complex and very expensive, making them unsuited for smaller satellites and cheaper missions.

Monocular cameras can fill this demand by being simple, cheap and generally light, also they are extremely proven as a space technology as they have been used on spacecrafts for decades.
Unfortunately, differently from the more complex stereo cameras, they do not give any information on the depth of the image, so, more complex image processing procedures must be implemented to estimate the full relative pose of the target spacecraft.

While one image can be used to accurately estimate the relative pose, this information is not enough to reconstruct the relative motion. This implies that a sequence of images has to be analyzed and its results need to be filtered in order to obtain the required data to enable close proximity operations.

The aim of this dissertation is to develop and test the aforementioned pipeline, starting from the neural network based pose estimation procedure devised by Massimo Piazza in his MSc thesis [Pia20].
To achieve this, two main components of the whole process had to be developed:

Firstly an image sequence generation algorithm that takes a physically accurate state propagation of the relative attitude between target and chaser and translates it to images. Secondly, a filtering step to post-process the data coming from the neural network.

1.2 State of the art

In this section, a brief overview of the state of the art regarding the core principles implemented in this dissertation will be pursued.

The three main topics treated in this section will be:

- *Space imagery generation*, how is currently tackled both through physical and virtual means.
- *Pose determination*, pros and cons of different algorithms for the estimation of the pose using images.
- *Filtering techniques*, an overview of the reasons for implementing them, and a brief description of the Kalman filter and its variations.

1.2.1 Space imagery generation

Lately, due to the increase in neural network applications in space systems, the need for several thousands images of orbiting satellites has arisen.
Unfortunately, the availability of real spaceborne images is quite limited, as few missions ever performed close proximity operations and collected such a volume of frames.
Moreover, the labeling procedure of each image is an extremely complex and time consuming task, especially considering that each neural network training has its specific labeling needs.
Due to this absence of data, two main alternative methods for obtaining spaceborne imagery have been developed. One physically based, and the second fully virtual.

The physically based platforms are generally composed by a set of directionable lights and numerically controlled cameras. These components are then mounted in a dark room with a high fidelity model of the considered spacecraft or planetary surface.
This setup manages to generate high fidelity spaceborne images, both of satellites and of surfaces (e.g. the lunar one) and, due to its numerically controlled nature, also attaches to each image a very accurate label of the relative pose considered.

One of the main facilities that uses physically based image generation is the TRON facility, managed by the SLAB team. It consists in a *10* x *4* m dark room with 10 Vicon Vero v1.3x cameras mounted on a controlled 6 DoF robotic arm that provides accurate positional data. Moreover the model is lighted by high accuracy light sources that mimic the sunlight and Earth's albedo. [KT08], [ESA21]

Figure 1.3: TRON facility (image credit: [AS21])

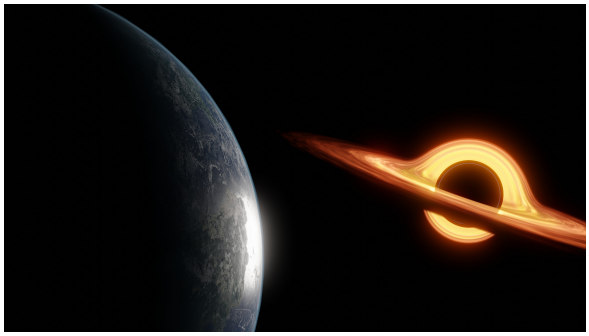
Instead, virtually based platforms generally exploit rendering softwares (such as Blender, Unreal engine 4, Octane Render...) to generate thousands of labeled images in a relatively short time span, and, thanks to their steady development through the years, photorealism can be achieved in a reasonable timeframe. Some of the main improvements can be attributed to: physically accurate ray tracing technology, more and better shading alternatives for textures, more complex shadow generation procedures and faster algorithms for the image generation.

Some of the main rendering softwares on the market to date are:

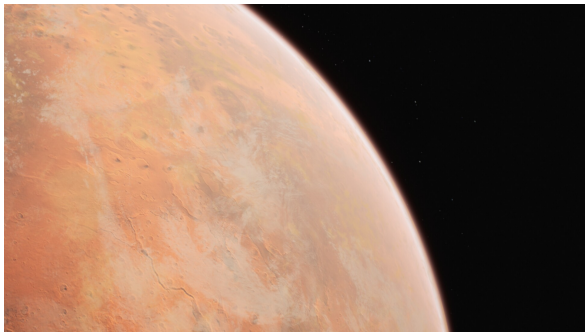
- *Blender*[1], an open source program with a huge community, mainly used for artistic renders, but, it has very good capabilities and can be considered for spaceborne imagery generation [Bla+21], [Sco+22].
- *Unreal Engine 5*[2], a very recent and still early access software, mainly used for creating videogames. However its real-time capabilities are impressive due to the sheer quality and photorealism of the frames. Its predecessor, Unreal Engine 4, has been already applied to the generation of spaceborne imagery through the URSO pipeline. [PG20]

Solutions specific to the aerospace industry are also present: those are specifically tailored to achieve the highest fidelity images for a wide array of space missions (e.g. planetary explorers, asteroid interception, close proximity operations between spacecrafts, ...).
Two of the main existing programs are ESA's PANGU [ESA19b] and Airbus SurRender [Air11], both based on the Open-GL platform: they have been used throughout the industry for hardware in the loop simulations for the development of various missions,

[1]The software can be found at: https://www.blender.org/
[2]The software can be found at: https://www.unrealengine.com/



(a) Blender render, (personal production)

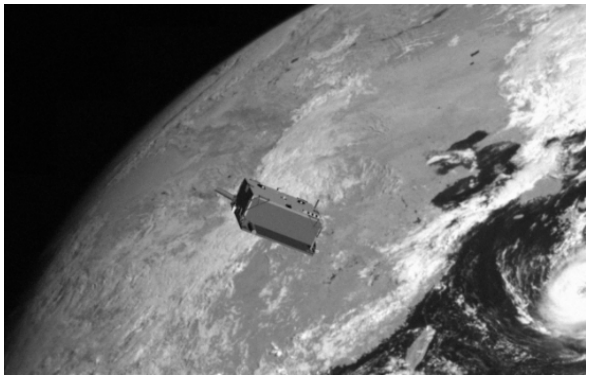


(b) Unreal Engine 5 render, (credits to: [Tor22])

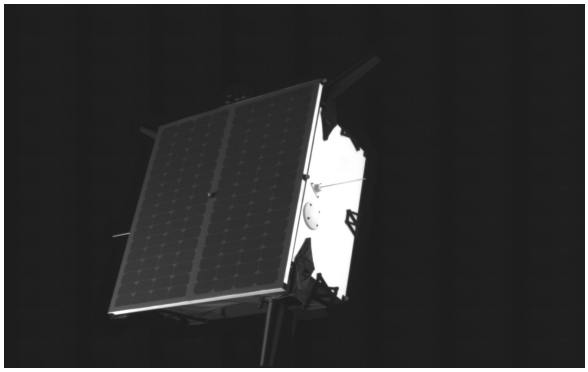
Figure 1.4: Example spaceborne landscapes obtained from Blender and Unreal Engine 5

such as JUICE, Exomars, MSR ERO... [Leb+21]

For the aim of this dissertation it is also important to focus on the SPEED database, as it will be extensively used as reference and comparison with the obtained results.
This dataset is based on the PRISMA mission, which was a technology demonstration mission launched in 2010 and used as a platform for testing formation flying and rendezvous technologies. It comprised of two spacecrafts, Mango, the chaser, and Tango, the target.
The SPEED database is composed of 15300 high fidelity images of the Tango spacecraft, of which 15000 are virtually generated and the remaining 300 have been physically obtained at the TRON facility.
It has been publicly released in 2019 as part of the Pose Estimation Challenge organized by SLAB in collaboration with ESA. In 2021 it received an expansion known as SPEED+ as part of the Pose estimation 2021 challenge, it now consists of 60000 synthetic images and 9531 real images [AS21].



(a) Virtually obtained image



(b) Physically obtained image

Figure 1.5: Example images from the SPEED database (image credit: [AS19])

As previously mentioned, the Real images have been obtained in the TRON facility where pictures of a high accuracy model of Tango have been taken using the same camera used by Mango (the chaser of the PRISMA mission). The whole dataset is accompanied by accurate labels, obtained thanks to the various controlled robotic arms deployed at the facility.
Instead the virtual portion of SPEED has been obtained from an OpenGL based pipeline where the target 3D model is placed at distances ranging from 3 m to 50 m, biased towards closer ranges, and presents a uniform attitude distribution.
Moreover the lighting closely matches the background, which is, for half of the database, made up of high resolution images of earth caught by the spacecraft Himawari-8 at different times and thus illumination conditions.

1.2.2 Pose determination

The aim of pose estimation algorithms is to determine the relative pose of a target spacecraft generally through visual means.
This process is achieved through complex image-processing softwares that identify the position and certain features on the target spacecraft, then a pose solver fits a known 3D model of the satellite to the aforementioned features. Doing so an estimated pose is obtained, generally with a good degree of accuracy.
As this method relies heavily on the visibility of the spacecraft features in the image, it is usable only during close proximity operations, so considering a target distance between a few centimeters and tens of meters.

These methods can be divided in two main subclasses depending on the approach chosen to tackle the problem, they are:

- *Feature-based methods*, which exploit the detection of high-level features in the image (e.g. straight edges, perfect circles, ...) to fit a 3D wireframe model of the spacecraft and determine its pose.
- *Deep learning-based methods*, they use neural networks to directly estimate the pose, or, identify known keypoints to then fit to a 3D model of the target.

Feature based methods are the traditional approach to computer vision based relative pose determination, as they have been widely developed and validated through the years.

They have been pioneered in 1989 with the main idea of detecting triplets of edges and matching them with a known 3D model to find its attitude [Dho+89].
This method firstly needs to define at least three edges on the target object and then

needs to solve an 8th degree equation to find a solution. Unfortunately this is quite a difficult task and is not computationally efficient, thus, improvements were proposed in the following years.

In 2014 a new method was proposed, it was based on the previous approach but was expanded to encompass more recent technologies and methodologies [DBJ14].
At first the image is filtered and features are highlighted: this is done through the use of a low pass filter, Canny edge detection and a Hough transform. These features are then grouped in "perceptual groups" which are also precomputed on a 3D wireframe model of the target, allowing to define an initial group of "most likely viewing directions". These will then be iteratively refined and the more promising ones are passed through steps of: *Newton-Raphson method*, *Model matching* and *Least Squares Fit* to obtain the final solution.
Although the precision of this method is very high, it is very computationally heavy and it has lower performance with more complex backgrounds.

In 2018 a new and improved method was proposed, the SVD algorithm, that offered a very accurate and fast method for relative pose determination [SVD18].
The main difference with respect to the previous one is the use of the "Weak Gradient Elimination" technique in parallel to the edge detection phase. This allows an increased robustness with respect to the background and less outliers, as the results of the parallel processes are properly merged.
As before, the detected features are grouped into higher level features to reduce the computational burden on the pose estimation, which now consists in an EPnP algorithm step and a *Newton-Raphson method* refinement. In the end the result that minimizes the reprojection error and satisfies a minimum error threshold is taken as the solution.
This method was then validated using real images from the PRISMA mission and showed very good performances making it capable of real time applications.

Moving on to deep learning based relative pose estimation methods, their application has been on the rise in the last decade. This can be partially attributed to recent improvements in the field of Neural Network based image processing algorithms, but also to the increase of available databases on which to train them.
A huge contribution can also be accredited to the "Relative pose estimation challenge" of 2019 which alone saw the contribution of 48 research groups and still accepts and evaluates "post-mortem" submissions [AS19].

These solutions generally make use of a Convolutional Neural Network (CNN) as it is the most versed solution to tackle an image processing problem.

It can perform different jobs in a NN pipeline depending on its aim, which can be:

- *Keypoint regression*: the pipeline finds notable features of the satellite from the image and uses a pose solver to define the attitude.
- *Direct pose estimation*: the pipeline directly estimates the pose of the spacecraft without relying on any pose solver.

Deep learning based solutions can be very precise, but their performance relies heavily on their training quality and time, as lack of insight from the programmer or insufficient training duration may lead to poor results. Furthermore they can be quite computationally intensive; nonetheless, being quite a recent and fast growing application, they can still be expected to heavily improve with time.

More in depth explanations on the inner workings of a Neural Network pipeline will be tackled in Section 2.2

1.2.3 Filtering techniques

Although relative pose estimation pipelines can be quite accurate, their raw measurements still present a certain degree of uncertainty. Thus, considering successive measurements, it can be noted that significant noise is present, rendering the use of filtering techniques very important.
Filters aim to accurately estimate the state at a certain time instant given a sequence of inaccurate and uncertain measurements. Consequently, they are a vital component on any spacecraft and, as such, they have to be robust and reliable but also computationally light.

A lot of different filtering techniques are present and are to be used for different measurement conditions, one such condition is the estimation of a state given a sequential set of measures.

As this is a common situation when developing space navigation systems, a wide arrangement of solutions is available such as the famous Kalman Filter (KF), which has become a staple of the industry [Kal60]. This can be attributed to its performance, versatility and attractive characteristics, such as the possibility to return estimates of hidden variables.

Although those are quite desirable properties, the simple Kalman Filter presents some evident flaws. Firstly, to be an optimal filter it needs to deal with a Gaussian, zero-mean, uncorrelated and white noise. Fortunately, it still retains good filtering characteristics if this assumption does not hold. Secondly, and most importantly, this filter can only be

used when the dynamics underlying the investigated phenomenon is linear. When the behavior of a system starts becoming non-linear and more complex, this kind of filters are no longer adequate.

To get around this problem some alternative variations have been developed over the years, the two most common being the Extended Kalman Filter (EKF) [MG02] and the Unscented Kalman Filter (UKF) [JUD95].

More specifically, the EKF solves the non-linearity problem by linearizing the system around the current state estimate. This is done through a *Taylor expansion* which can be formulated to include higher order terms in order to maintain a good filtering behaviour for highly non-linear systems. The main issues with the EKF is that it can't easily deal with the aforementioned strong non-linearities and is difficult to properly tune.

Instead, the UKF tackles this issues in a different way, by employing an Unscented Transform (UT) to propagate the mean and covariance of the state. Thanks to the higher performance of the aforementioned transformation with respect to the linearization, the UKF is also better suited than the EKF to deal with highly non-linear problems. One further advantage is also given by the absence of a Jacobian to be computed as is done in the EKF.
Unfortunately this filter is quite computationally demanding, as an increased number of propagations is needed due to the UT, making it significantly slower than its alternative.

A more in depth analysis of the Kalman filtering techniques, both linear and non-linear is provided in Section 3.4

1.3 Outline

This dissertation is composed of 5 chapters:

- *Chapter 2*: dives into the basic theoretical knowledge necessary to understand this dissertation, such as: relative pose, deep learning applied to image processing, filtering techniques.
- *Chapter 3*: presents the work done in detail, in particular: the image generation procedure, the sequence generation, the filtering pipeline.
- *Chapter 4*: analyzes the results obtained from the proposed pipeline.
- *Chapter 5*: summarizes the results obtained and gives some recommendations for future developments.

2 | Theoretical background

2.1 Relative pose fundamentals

This section will briefly present the core concepts related to the pose determination of a satellite and the dynamics underlying its translational and angular motion using mostly equations from [MG00].
The attitude of an object can be described as the relative orientation between the body centered reference frame of the object and another reference frame. This property can be represented with a rotation matrix, generally known as Direction Cosine Matrix (DCM). This rotation can be mathematically formalized as:

$$\boldsymbol{B} = \boldsymbol{A}_{B/N}\boldsymbol{N} \tag{2.1}$$

Where $\boldsymbol{B}$ represents the body fixed reference frame, $\boldsymbol{N}$ represents the alternative reference frame and $\boldsymbol{A}_{B/N}$ is the DCM. It is also important to note that, thanks to the attitude matrix being orthogonal, the inverse rotation can be easily achieved as $\boldsymbol{A}_{N/B} = \boldsymbol{A}_{B/N}^T$.
Considering matrix algebra, it is possible to obtain an eigenvector ($\underline{e}$) from any real, orthogonal matrix, which displays the property:

$$\underline{e} = \boldsymbol{A}\underline{e} \tag{2.2}$$

This can be seen as if vector $\underline{e}$ does not change due to the rotation of matrix $\boldsymbol{A}$, as such the rotation must be performed around axis $\underline{e}$. This theorem is known as *Euler's rotation Theorem* and, correspondingly, the rotation axis is known as *Euler axis* and the rotation angle applied in this transformation is known as *Euler angle* (ϕ).
A peculiar property of the aforementioned parametrization is that the rotation $[\phi, \underline{e}]$ is virtually equivalent to the rotation $[-\phi, -\underline{e}]$. This adds versatility to the derived alternative notations and will be leveraged in the later parts of the dissertation.
Euler's parametrization, although quite simple and having an evident physical meaning, presents a critical drawback as a singularity is present when ϕ coincides with $n\pi$: in this case $\underline{e}$ is not uniquely defined.

2.1.1 Quaternions and MRPs

Quaternions provide one of the most useful notation for representing spatial orientations of an object in space, as such they are also commonly used through a wide array of different industries.
They are defined as a unitary norm vector of four components, linked to the Euler axis and angle as:

$$\begin{cases} q_0 = \cos(\frac{\phi}{2}) \\ q_1 = e_1 \sin(\frac{\phi}{2}) \\ q_2 = e_2 \sin(\frac{\phi}{2}) \\ q_3 = e_3 \sin(\frac{\phi}{2}) \end{cases} \tag{2.3}$$

Note that the notation is made up of three vectorial components (q_1, q_2, q_3) and one scalar component (q_0). The notation used in this dissertation imposes the scalar component as the first of the quaternion and is called *WXYZ notation*.

Differently from Euler angles the quaternions do not present any singularity and are generally easier to implement, but, they lose the evident physical meaning of the Euler notation leading to less intuitive results. Furthermore, a critical drawback is the need to keep the quaternion vector norm unitary to ensure it actually represents a rotation. This means that, for filters, special measures have to be implemented to avoid additive errors such as in MEKF [LMS82].

Due to this critical drawback, as the attitude parametrization chosen will impact the dynamics and thus the filtering procedure, an alternative notation will be exploited in this dissertation.

The Modified Rodrigues Parameter (MRP) [SJ95] parametrization was chosen as its simplicity and qualities proved desirable features. MRPs can be easily derived from quaternions or Euler notation as:

$$\zeta = \frac{\tilde{\boldsymbol{q}}}{1 + q_0} = \underline{e} \tan\left(\frac{\phi}{4}\right) \qquad \text{where} \qquad \tilde{\boldsymbol{q}} = [q_1, q_2, q_3] \tag{2.4}$$

Unfortunately one evident singularity is present: in fact it can be seen that, as ϕ approaches 2π rad, the norm of the MRP tends to infinity. This is different from the Euler notation, where the same happened every $n\pi$ rad.

The presence of a singularity can pose a huge problem as small variations in the attitude near this point may lead to extremely big variations in the MRPs.
Fortunately a way to avoid both these problems has been devised in the Shadow MRP set, which leverages the fact that the attitude identified as $[q_0, \tilde{\boldsymbol{q}}]$ is the same as $[-q_0, -\tilde{\boldsymbol{q}}]$. Consequently the shadow MRP set has been defined as:

$$\zeta_S = \frac{-\tilde{\boldsymbol{q}}}{1 - q_0} = \underline{e}\, tan\Big(\frac{\phi - 2\pi}{4}\Big) \tag{2.5}$$

This alternative set moves the singularity at 0 rad, effectively flipping the behaviour of the regular MRP set while retaining the same attitude information. It can also be noticed that in the points where $\phi = [\pi, -\pi]$ the norm of both Shadow and regular set is the same and equal to 1.
Consequently, to fully and accurately describe the attitude of a target, the two sets can be exploited as:

$$\begin{aligned} &\text{if norm}(\zeta) > 1 \rightarrow \zeta_S \text{ is used to describe the attitude} \\ &\text{if norm}(\zeta_S) > 1 \rightarrow \zeta \text{ is used to describe the attitude} \end{aligned}$$

Unfortunately to avoid the singularity two discontinuities have to be introduced, but, this is deemed as an acceptable drawback to allow a good description of the attitude.

An alternative way to describe this process is to geometrically visualize the problem as a simplified 2D stereographic projection.

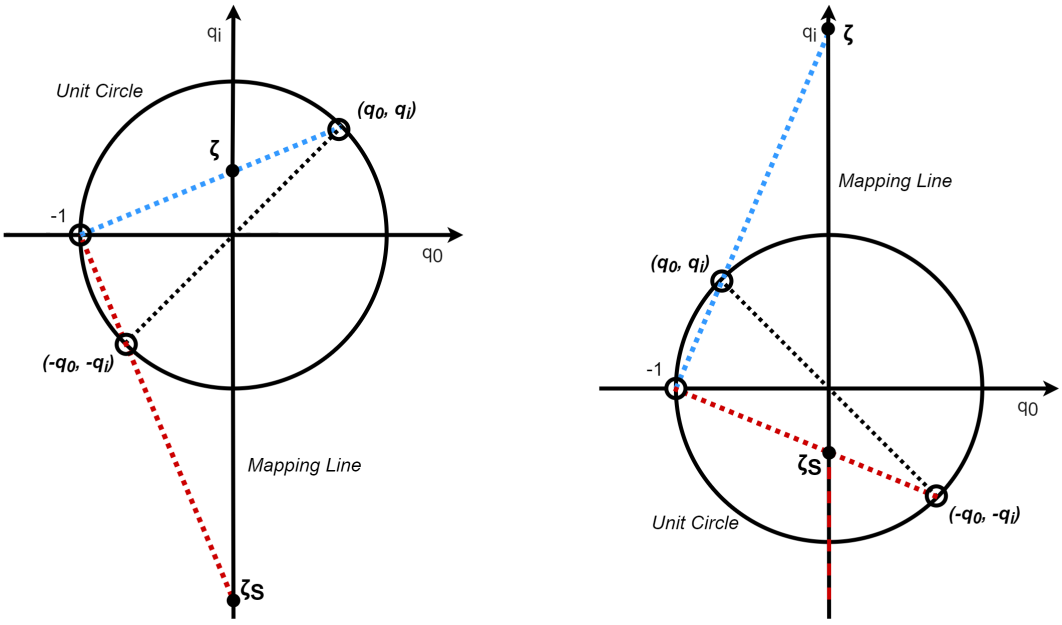


Figure 2.1: Visualized MRPs on a simplified stereographic projection

Figure 2.1 represents a $[q_0, q_i]$ space (where q_i is an alternative notation for $\tilde{\boldsymbol{q}}$), on which both the considered attitudes are expressed in quaternions. It is important to note that, they lie on the unit circle as their norm is unitary.
The *y axis* is considered the "Mapping line" meaning that the MRPs associated with the considered attitudes will be projected there.
Two different situations have been visualized. Firstly a situation where ϕ lies within $[-\pi, \pi]$, thus, the choice of the regular MRP (ζ) is advantageous as it lies within the unit circle and has a norm smaller than one.
Considering the second case, now the attitude changed and ϕ moved within $[\pi, -\pi]$, as such it is evident that the Shadow MRP (ζ_S) is more favorable as the regular MRP started to diverge.

It is important to highlight that the presented example is a very simplified model, as in actuality the quaternions lie on a "Unit sphere" and the stereographic projection is performed from a sphere to a hyperplane.

2.1.2 Relative distance problem

The relative attitude problem is composed of two main components, translation and pose determination. This dissertation will treat them separately.

Starting from the relative distance problem, it aims to find the evolution of both relative separation between the target and the chaser and the associated relative velocity.
All the equations are expressed in the Local Vertical Local Horizontal (LVLH) frame visualized in Figure 2.2.
Consequently the relative parameters can be expressed as:

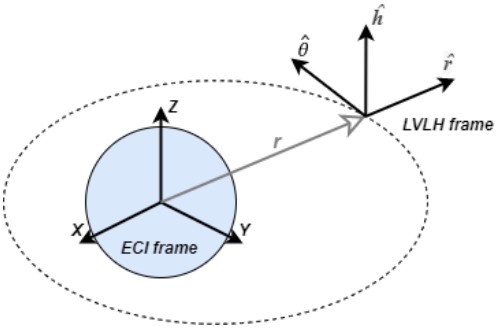


Figure 2.2: LVLH frame visualization

$$\mathbf{r_r} = x\hat{\boldsymbol{r}} + y\hat{\boldsymbol{\theta}} + z\hat{\boldsymbol{h}} \tag{2.6}$$

$$\mathbf{v_r} = \dot{x}\hat{\boldsymbol{r}} + \dot{y}\hat{\boldsymbol{\theta}} + \dot{z}\hat{\boldsymbol{h}} \tag{2.7}$$

Where $\mathbf{r_r}$ is the relative distance and $[x, y, z]$ are its vector components, $\mathbf{v_r}$ is the relative velocity and $[\hat{\boldsymbol{r}}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{h}}]$ are the versors of the considered LVLH triad.
Focusing on the relative translational dynamics equations, they describe the evolution of the components of the relative distance. Note that the orbit chosen and the initial

separation between the two spacecrafts is extremely important and heavily impacts the evolution of the dynamics.
This problem can be expressed with the equations [SJ03]:

$$\begin{cases} \ddot{x} = 2\dot{\nu}\dot{y} + \ddot{\nu}y + \dot{\nu}^2 x + \dfrac{\mu}{\bar{r}^2} - \dfrac{\mu(\bar{r}+x)}{[(\bar{r}+x)^2 + y^2 + z^2]^{3/2}} \\ \ddot{y} = -2\dot{\nu}\dot{x} - \ddot{\nu}x + \dot{\nu}^2 y - \dfrac{\mu y}{[(\bar{r}+x)^2 + y^2 + z^2]^{3/2}} \\ \ddot{z} = -\dfrac{\mu z}{[(\bar{r}+x)^2 + y^2 + z^2]^{3/2}} \end{cases} \tag{2.8}$$

Where μ is the standard gravitational parameter of the main attractor, ν is the true anomaly and $\bar{r}$ represents the distance between the main attractor center and the chaser center.
To complete the problem, the motion of the chaser spacecraft must be described both in terms of true anomaly (ν) and position ($\bar{r}$), this can be done as:

$$\ddot{\bar{r}} = \bar{r}\dot{\nu}^2 - \frac{\mu}{\bar{r}^2} \tag{2.9}$$

$$\ddot{\nu} = -\frac{2\dot{\bar{r}}\dot{\nu}}{\bar{r}} \tag{2.10}$$

Solving this problem provides a good representation of the chaser-target relative distance evolution. It is important to keep in mind that in a close proximity operations scenario, the dynamics can be quite slow and thus, weakly nonlinear.

2.1.3 Relative attitude problem

Focusing now on the relative attitude, it presents a more complex problem with a faster evolution.

The general equations governing the attitude motion of a rigid body in its body frame are known as *Euler equations*. They have the form:

$$\mathbf{J}\dot{\omega} + \omega \times \mathbf{J}\omega \ = \mathbf{M} \tag{2.11}$$

Where $\mathbf{J}$ represents the object inertia matrix, ω is its angular velocity vector, and $\mathbf{M}$ is the sum of the applied torques.

The procedure followed in this dissertation needs to put in place a key assumption: the chaser attitude is already perfectly known and can be described using the *Torque free*

Euler equations ($\mathbf{M} = \underline{0}$). This is assumed reasonable as on one hand, in a real scenario, the chaser spacecraft has its own pose determination subsystem and knows its attitude with a low degree of uncertainty. On the other hand, in a simulated environment, torque free dynamics can be assumed as their propagation is carried out over a very small time frame and thus the disturbing effects are close to negligible.

For the aim of the relative attitude problem, at first there is the need to contextualize the present angular velocities and their relations. This is done as [MZ14]:

$$\begin{aligned} \omega_r &= \omega_t - \mathbf{\Gamma}\omega_c \\ \dot{\omega}_r &= \dot{\omega}_t - \mathbf{\Gamma}\dot{\omega}_c + \dot{\omega}_{app} \\ \dot{\omega}_{app} &= \omega_r \times \mathbf{\Gamma}\omega_c \end{aligned} \tag{2.12}$$

Where ω_t and ω_c are respectively the target and the chaser angular velocities expressed in their body fixed reference frame and ω_r represents the relative angular velocity expressed in the target body frame. Finally $\mathbf{\Gamma}$ is the rotation matrix linking the body-fixed reference frame of the target to the body-fixed reference frame of the chaser.
The aforementioned matrix can be attained from the MRPs vector(ζ) following the relation:

$$\mathbf{\Gamma}(\zeta) = \boldsymbol{I}_3 - \alpha_1^A[\zeta\times] + \alpha_2^A[\zeta\times]^2 \tag{2.13}$$

Where $\boldsymbol{I}_3$ is the $[3 \times 3]$ Identity matrix and the parameters $\alpha_1^A, \alpha_2^A, [\zeta\times]$ can be obtained as:

$$\begin{cases} \alpha_1^A = 4\dfrac{1 - \zeta^T\zeta}{(1 + \zeta^T\zeta)^2} \\ \alpha_2^A = 8\dfrac{1}{(1 + \zeta^T\zeta)^2} \\ [\zeta\times] = \begin{bmatrix} 0 & -\zeta_3 & \zeta_2 \\ \zeta_3 & 0 & -\zeta_1 \\ -\zeta_2 & \zeta_1 & 0 \end{bmatrix} \end{cases} \tag{2.14}$$

The evolution of the attitude matrix can then be linked to the dynamics of the relative MRPs, and the variation of the relative angular velocity can be described by properly

manipulating the Euler equation of the target accounting for the kinematic relationships found before in Equation 2.12.
Thus a set of equations describing the evolution of the system can be attained as:

$$\begin{cases} \dot{\zeta}_r = \frac{1}{4}\Sigma(\zeta_r)\omega_r \\ \boldsymbol{J_t}\dot{\omega}_r + \omega_r \times \boldsymbol{J_t}\dot{\omega}_r = \boldsymbol{M}_{app} - \boldsymbol{M}_g - \boldsymbol{M}_{ci} \end{cases} \tag{2.15}$$

where $\boldsymbol{J_t}$ is the inertia matrix of the target, ζ_r is the MRP vector describing the relative attitude and:

$$\begin{gathered} \Sigma(\zeta) = (1 - \zeta^T\zeta)\boldsymbol{I}_3 + 2\zeta\zeta^T + 2[\zeta\times] \\ \boldsymbol{M}_{app} = \boldsymbol{J_t}\dot{\omega}_r \times \Gamma\omega_c \\ \boldsymbol{M}_g = \Gamma\omega_c \times \boldsymbol{J_t}\Gamma\omega_c + (\omega_r \times \boldsymbol{J_t}\Gamma\omega_c + \Gamma\omega_c \times \boldsymbol{J_t}\omega_r) \\ \boldsymbol{M}_{ci} = \boldsymbol{J_t}\Gamma\dot{\omega}_c \end{gathered} \tag{2.16}$$

where $\boldsymbol{M}_{app}$ represents the apparent torques, $\boldsymbol{M}_g$ are the gyroscopic torques and $\boldsymbol{M}_{ci}$ are the chaser-inertial torques.

It is important to notice that the relative pose dynamics and its evolution heavily depend on the angular velocity at which the target is tumbling. The faster the tumbling, the faster the dynamics, and thus, the higher the non-linearity of the problem.

2.2 Relative Pose Estimation Pipeline

This dissertation exploits the machine learning based Relative Pose Estimation Pipeline developed by Massimo Piazza in his Master Degree thesis [Pia20],
It mainly consists of three stages: firstly the image is ran through SLN which recognizes where TANGO is in the frame, secondly the image is cropped and fed through LRN which identifies the notable keypoints and gives their position to the next stage, Pose Estimation which estimates the relative pose given a 3D wireframe model.
More in depth information can be found in the following sections.

2.2.1 Introduction to NN and CNN

Over the past few years Neural Networks (NN) have been in the spotlight for their impressive capabilities and versatility and, thanks to this, they have been used for a wide array of applications, from aerospace to medicine.
NN are a computational model loosely based on a biological brain, as they exploit interconnected nodes called artificial neurons connected through synapses.

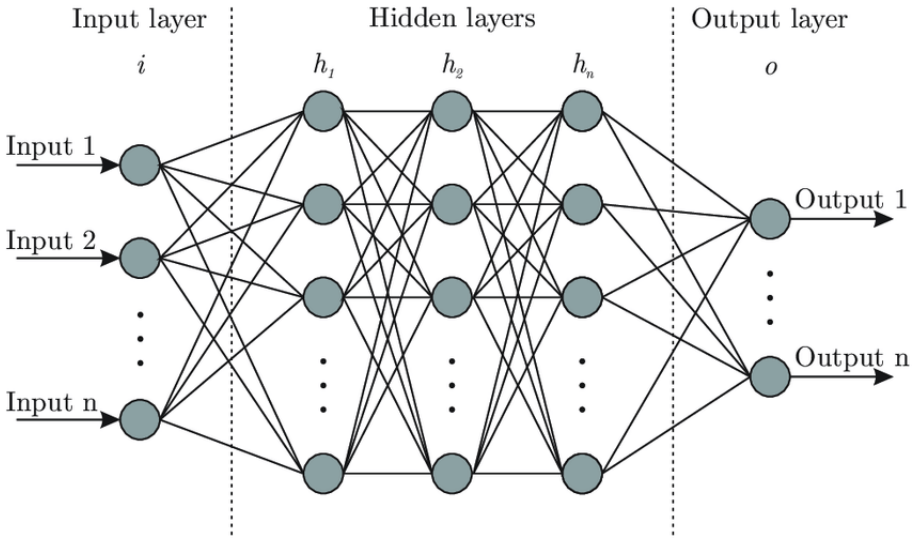


Figure 2.3: Neural Networks (NN) sample architecture, (image credit: [BGF17])

Artificial neurons are organized in layers, the first being the input layer, followed by several hidden layers, where the true data processing occurs, and finally an output layer where the result is displayed. Each layer is connected to the next one by a set of weighted synapses, which increase or decrease the strength of the next connected neuron.

Training is the process that allows the NN to draw useful information from the inputs and give coherent outputs.

To train a NN a vast amount of data is needed and often, if a supervised learning process is chosen, Ground Truth (GT) labels are necessary for every single sample. Additionally, when creating the training set it is very important to be aware of potential biases, as they are very likely to translate onto loss of performance or unexpected outputs from the NN. After the definition of a coherent and useful data set, the training can take place.
Initially each synapse weight can be imposed randomly or can be based on pre-trained NN weights in order to shorten the process (i.e., transfer learning). There are many different kinds of training procedures, which are all based on evaluating the discrepancy between the output predicted by the NN and the desired output. This difference is then used to adjust the weights connecting the various layers according to different training rules. After an extensive training stage the NN can reliably perform the task it was trained to do.

Many different types of NN are available nowadays, all with their respective strengths and weaknesses. Three main typologies can be considered:

- *Artificial Neural Network (ANN)*, the less complex one being the simple input-output processes already described.
- *Recursive Neural Network (RNN)*, widely used for the study of sequential input patterns (e.g. processes evolving in time). This sequences can be recognized and determined because the state of each neuron is influenced by its state at the previous iteration, allowing the network to achieve a sort of memory of the process.
- *Convolutional Neural Network (CNN)*, widely used for 2D problems. They can leverage operations such as convolution and pooling to process spacial features such as those found in images.

Focusing on the latter, CNN have seen significant developments in the last years especially in the image and video processing field.
They mainly work by extracting simple features of an image and then assembling them into higher level features, while retaining spatial information. The latter are then combined and thus can be attributed to an output.
A simple and intuitive example could be the recognition of a hand written number: at first an image is loaded and lower level features can be extracted, such as straight lines or arcs. Secondly this lower level features can be grouped into higher level ones to recognize patterns, for example circles or groups of segments. Finally, based on the detected patterns a guess can be made: for example, two circles may lead to a 8.

These processes are achieved by two fundamental operations of the CNN:

- ***Convolution*** is a linear operation that multiplies a set of weights with the input. This operation is performed using a two-dimensional array of weights, called a filter or a kernel, that slides through the input volume with a certain stride.
  According to the notation of Figure 2.4 the green slot on the output layer is obtained by multiplying the dark blue region on the input layer for the kernel. Following this operation the blue box is moved one slot to the right and an analogous process can be performed.

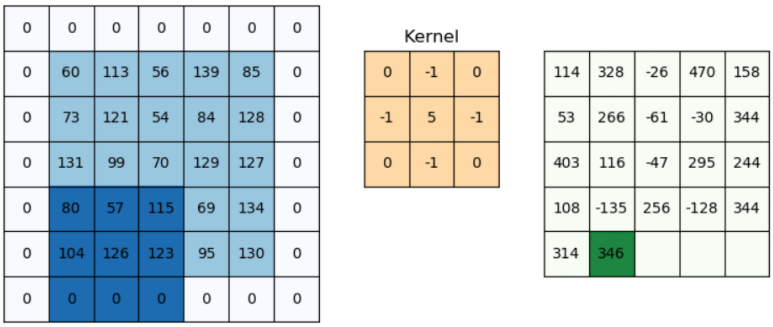


Figure 2.4: Example image of the convolution operation

- ***Pooling*** consists in either taking the maximum or the average (max or average pooling) of the elements enclosed in a certain fixed window of the input volume. This operation aims to reduce the volume of successive layers and increase the CNN robustness. Generally the size of a pooling window is 2-3 pixels.

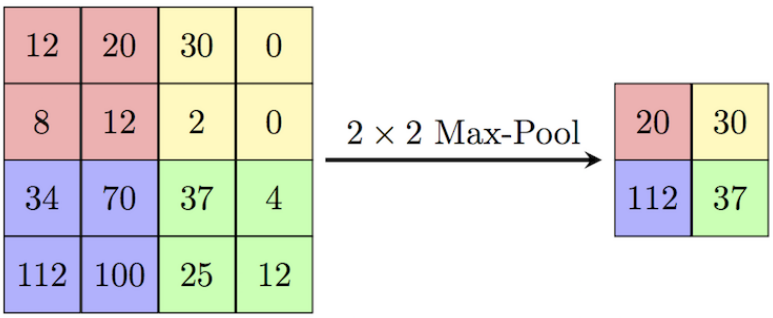


Figure 2.5: Example image of the Pooling operation

CNNs are made up of three main types of layers: Convolutional layers, Pooling layers and Fully connected layers.
The convolutional and pooling layers usually alternate in order to condense the input information: they are the layers responsible for the feature extraction and processing. Afterwards, the information obtained is wrapped into a vector and passed through a set

of Fully connected layers (analogous to the layers of a ANN) to allow the classification of the input image according to a set of predetermined classes.

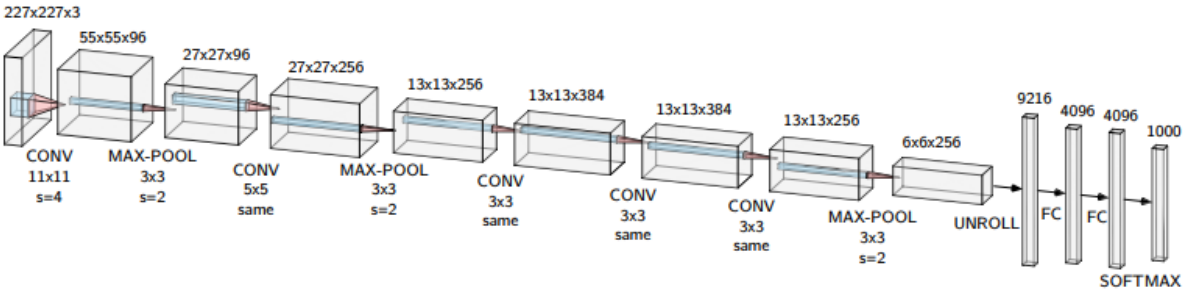


Figure 2.6: Example image of the *Alexnet* CNN (Image credit: [Pia20])

A notable example of CNN is the open source You Only Look Once (YOLO) object detection model [Red+16].

Figure 2.7: Example of YOLO v5 (image credit: [ult20])

This algorithm manages to detect an object class as well as its Bounding Box (BB) and can do so for multiple objects in an image. It is relatively light managing to reach real-time image processing at 45 frames per second on common off the shelf GPUs. Finally it also manages great performances with its simpler yet effective architecture.

2.2.2 SLN, LRN and EPnP

The RPEP devised by Massimo Piazza is capable of giving an accurate estimate of the relative pose of a target spacecraft, from a single monocular grayscale image. This feat is achieved by daisy chaining two CNNs which have been selected and trained in order to fulfill specific purposes.
The pipeline can be divided into three core sections with different tasks:

- *Spacecraft Localization Network (SLN)*: It leverages the YOLOv5s CNN architecture [ult20] to define the BB of the spacecraft
- *Landmark Regression Network (LRN)*: It leverages the HRNet CNN architecture [Sun+19] to identify and localize certain features of the target spacecraft

- *Pose Solver*: It uses analytical techniques to fit a known 3D wireframe model to the features identified previously and thus find the pose

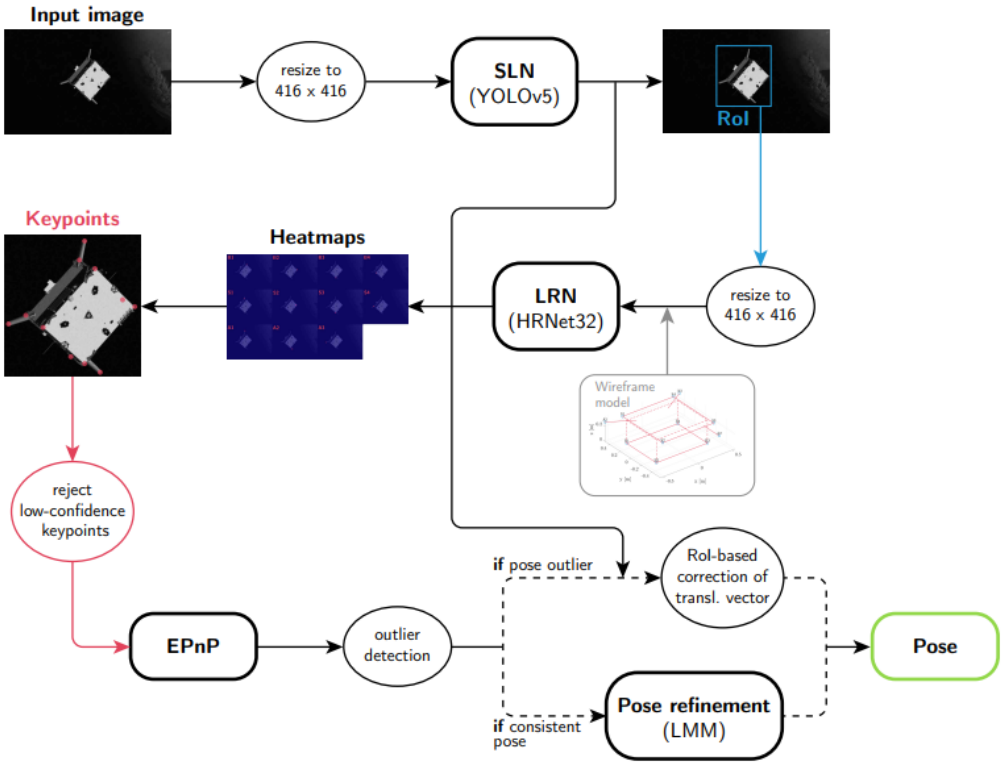


Figure 2.8: Architecture of the RPEP, (image credit: [Pia20])

The description of each submodule starts from Spacecraft Localization Network (SLN), the first image processing step of the RPEP.
It is based on the YOLOv5s architecture (the smallest version of YOLOv5), due to its favorable trade-off between computational efficiency and accuracy.
The CNN receives as input a properly resized image of the target, processes it and returns the coordinates of the detected BB.

During training and testing the pipeline ran on images extracted from the SPEED database, which presented a number of frames with Earth as background. This was aimed at improving the pipeline robustness to noise and background and, as can be seen from Figure 2.9 this was achieved as the pipeline is extremely effective, managing to detect the spacecraft even in cases where the human eye may fall short.

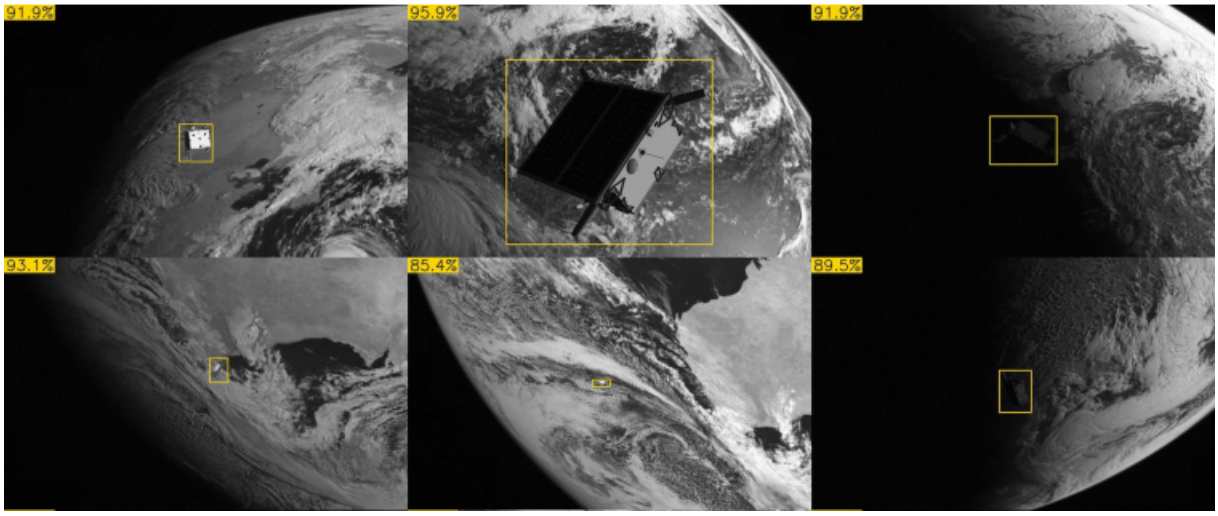
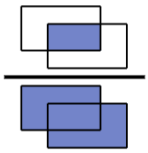


Figure 2.9: SLN prediction on 6 test images with Earth background, (image credit: [Pia20])

The pipeline was then tested and the results were noted using the Intersection over Union (IoU) metric, which defines the overlap ratio between the real and predicted BB by dividing their intersection (the area they have in common) by their union (the sum of their respective areas).

$$IoU = \frac{BB_{GT} \cap BB_{Pred}}{BB_{GT} \cup BB_{Pred}} = \quad (2.17)$$



Overall the pipeline managed to achieve a mean IoU of *0.9538* with median IoU of *0.9650* on the whole SPEED dataset. This result outperformed UniAdelaide [Che+19], the winner of the Pose Estimation Challenge proving how effective the SLN is.

Moving forward, the Landmark Regression Network (LRN) is the second step of the pipeline. It is responsible for the detection of the notable features of the spacecraft in the considered frame.
HRNet32 [Sun+19] has been chosen as the basic CNN architecture of this stage thanks to its unprecedented accuracy in the field of human pose estimation. Also the trade-off between computational efficiency and accuracy has been considered, driving the choice of HRNet32 over HRNet48.

A rescaled image of the BB found previously is given as an input to HRNet, which has been trained to regress heatmaps of these frames, highlighting the identified features as peaks in each heatmap. It is also important to note that the CNN is able to accurately predict the position of keypoints occluded by a portion of the spacecraft itself.

Overall eleven keypoints were considered: three corresponding to the tip of each antenna (A1, A2, A3), four corresponding to the edges of the solar panel (S1, S2, S3, S4), and the remaining four related to the edges of the backplate of the spacecraft (B1, B2, B3, B4).

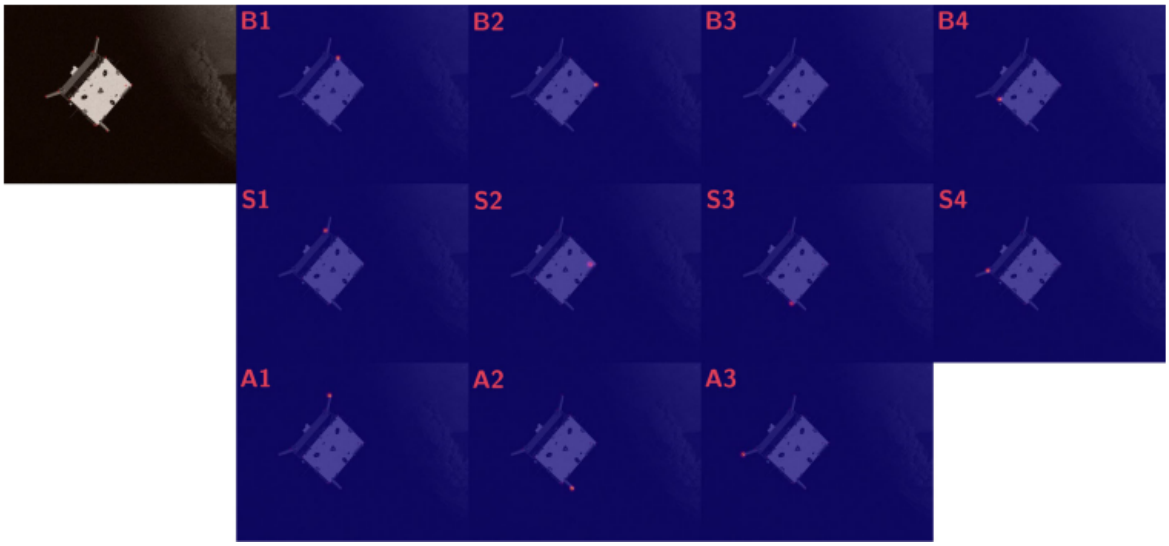


Figure 2.10: Heatmap regressed by HRNet, (image credit: [Pia20])

Although an excellent regression accuracy had been achieved, the LRN proved to be the least computationally efficient stage of the pipeline, around one order of magnitude slower than SLN.
Unfortunatelly this happens because the size of HRNet is bigger than its YOLO counterpart, and thus a higher number of operations has to be performed.

To conclude this overview, the Pose Solver is the last stage of the RPEP and its task is to predict the relative pose of the target spacecraft.
This is achieved through analytical means given the BB and the detected keypoints, as illustrated in Massimo Piazza's work [Pia20].

The first step of this process is the elimination of the lower confidence landmarks as the EPnP algorithm is not robust to outliers and, even a small number of them, may lead to incorrect pose estimates. To this avail it was preferred to work on a smaller sample of high confidence keypoints instead of working with the whole population.

After keypoint rejection the remaining data is fed through the EPnP algorithm which, without requiring any initial guess, returns the predicted pose.

The pose is then compared to the BB. If it is coherent with it, the result is deemed correct and is refined using the Levenberg-Marquardt Method [Gav13]. Instead if the

pose is deemed uncoherent with the detected BB, a lower accuracy estimate is given based on the detected BB.

2.2.3 Performance metrics

Overall the Relative Pose Estimation Pipeline devised by Massimo Piazza managed to achieve great results, still standing at 5th place on the post mortem SLAB/ESA Pose Estimation Challenge [SLA19] with a 3rd place all time best.
Table 2.1 summarizes the pipeline performances in terms of translational (E_t) and rotational absolute error (E_θ) and their corresponding standard deviation.

Absolute error		
	Mean	***Median***
E_t	*10.36 cm*	*3.58 cm*
$\boldsymbol{E_t}$	*[0.52, 0.56, 10.25] cm*	*[0.24, 0.27, 3.50] cm*
E_θ	*2.24°*	*0.81°*
$\boldsymbol{E_\theta}$	*[1.57°, 0.84°, 1.72°]*	*[0.52°, 0.33°, 0.34°]*
Standard Deviation		
$\boldsymbol{\sigma_{E_t}}$	*[1.62, 1.71, 30.44] cm*	
$\boldsymbol{\sigma_{E_\theta}}$	*[8.92°, 5.11°, 10.82°]*	

Table 2.1: Global end-to-end performance of the RPEP

As can be seen performance in the order of centimeters and degrees level is achieved, as such the pipeline can be seen as compatible with close proximity operations.
It is important also to point out that accuracy is assessed over the whole SPEED dataset, but, generally, the error decreases greatly as the target gets closer.

2.3 Kalman filter basics

In the space industry, the most used and famous filters are the Kalman Filter (KF) and its variants [Kal60]. They are very efficient recursive filters: in fact they use the output of the previous iteration as an initial estimate for the next one.
The basic form of the KF is very powerful for dealing with linear problems, whereas its variants are needed for dealing with more complex non-linear problems.

In this section the basic form of the KF will initially be presented and then its definition will be expanded to include also the EKF and the UKF.

Considering a linear discrete-time system in the form:

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{u}_k + \mathbf{w}_{(k)} \\ \mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \end{cases} \tag{2.18}$$

Where $\mathbf{x}_k$ and $\mathbf{u}_k$ represent respectively the state and the input at timestep k, instead, $\mathbf{w}_k$ and $\mathbf{v}_k$ are used to identify the process noise and the measurement noise, whereas $\mathbf{y}_k$ identifies the measurement output at timestep k. Finally the matrices $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{H}$ respectively represent the State Transition Matrix (STM), the Input transition matrix (ITM) and the Observation matrix.

This equation can be used during the *Prediction step* of the filter, where, knowing the mean value of the state estimation at the previous iteration $\hat{\mathbf{x}}_k^+$, a propagation can be performed as:

$$\hat{\mathbf{x}}_{k+1}^- = \mathbf{F}\hat{\mathbf{x}}_k^+ + \mathbf{G}\mathbf{u}_k \tag{2.19}$$

Which returns the expected mean value of the state $\hat{\mathbf{x}}_{k+1}^-$ for the current iteration.
In a similar fashion the covariance matrix at the previous iteration $\mathbf{P}_k^+$ can be propagated to obtain its expected value $\mathbf{P}_{k+1}^-$ at the current timestep, this is done as:

$$\mathbf{P}_{k+1}^- = \mathbf{F}\mathbf{P}_k^+\mathbf{F}^T + \mathbf{Q}_k \tag{2.20}$$

Note that $\mathbf{Q}_k$ represents the covariance matrix associated to the process noise, while $\mathbf{R}_k$ is associated to the measurement noise.
The next step is defined as *Correction step* as it corrects the predicted state based on the latest measurements available.

To this avail, a *Kalman gain* $\mathbf{K}_{k+1}$ is defined and has the aim of connecting and weighting the predicted state and the incoming measurements, giving more importance to the less uncertain one.

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1}^{-}\mathbf{H}^{T}(\mathbf{H}\mathbf{P}_{k+1}^{-}\mathbf{H}^{T} + \mathbf{R}_{k+1})^{-1} \tag{2.21}$$

Once the *Kalman gain* has been computed, it is used to update the state and the covariance matrix as:

$$\hat{\mathbf{x}}_{k+1}^{+} = \hat{\mathbf{x}}_{k+1}^{-} + \mathbf{K}_{k+1}(\mathbf{y}_{k+1} - \mathbf{H}\hat{\mathbf{x}}_{k+1}^{-}) \tag{2.22}$$

$$\mathbf{P}_{k+1}^{+} = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H})\mathbf{P}_{k+1}^{-} \tag{2.23}$$

Note that the output of the *Correction step* becomes the input of the next filter iteration as the filter is a "predictor-corrector" type and uses recursive information to improve its guesses.
As previously stated and clearly visible from the dynamics taken into consideration, the simple KF is only usable for linear conditions. However, the vast majority of the phenomena considered in the industry is non-linear and often can not be approximated with linear dynamics.
In this case different solutions must be adopted and non-linear estimators have to be employed.

2.3.1 EKF

One of the most prevalent solutions to tackle the non-linear state estimation problem is the Extended Kalman Filter (EKF) [MG02], a modified version of the standard KF.
By linearizing the non-linear model around the state estimate of the previous iteration, and using that to estimate the state, the filter manages to overcome the linearity problem of the KF.
In this section the discrete-time EKF is presented starting from the non-linear model:

$$\begin{cases} \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \\ \mathbf{y}_k = h(\mathbf{x}_k, \mathbf{v}_k) \end{cases} \tag{2.24}$$

To perform the linearization step the computation of two Jacobians is needed: the first $\mathbf{F}$ related to the state transition function $f(\cdot)$ and the second $\mathbf{H}$ related to the measurement

function $h(\cdot)$.
As the linearization is effectively based on a *Taylor expansion*, higher order EKF can be obtained by retaining more terms during the procedure.

$$\mathbf{F} = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^+} \qquad \mathbf{H} = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^-} \tag{2.25}$$

Once the Jacobians have been computed, the filter plays out similarly to a KF, with the main difference being in the use of the matrices $\mathbf{H}$ and $\mathbf{F}$ obtained through the linearization procedure.
Here the steps to be performed are reminded:

$$\hat{\mathbf{x}}_{k+1}^- = f(\hat{\mathbf{x}}_k^+, \mathbf{u}_k, \mathbf{0}) \tag{2.26}$$
$$\mathbf{P}_{k+1}^- = \mathbf{F}\mathbf{P}_k^+\mathbf{F}^T + \mathbf{Q}_k \tag{2.27}$$
$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1}^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_{k+1}^-\mathbf{H}^T + \mathbf{R}_{k+1})^{-1} \tag{2.28}$$
$$\hat{\mathbf{x}}_{k+1}^+ = \hat{\mathbf{x}}_{k+1}^- + \mathbf{K}_{k+1}(\mathbf{y}_{k+1} - h(\hat{\mathbf{x}}_{k+1}^-, \mathbf{0})) \tag{2.29}$$
$$\mathbf{P}_{k+1}^+ = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H})\mathbf{P}_{k+1}^- \tag{2.30}$$

The EKF is proven to be quite capable and efficient, but, presents a few drawbacks.
It is generally difficult to tune as it relies on linearization and it has problems when dealing with systems that present significant non-linearities.
These drawbacks create the need for a more reliable filter when dealing with such systems, fortunately the UKF fills this niche.

2.3.2 UKF

The Unscented Kalman Filter (UKF) [JUD95] overcomes the drawbacks of the EKF by introducing three tuning parameters and avoiding the linearization completely, opting to rely on an Unscented Transform (UT) for the propagation of the mean and the covariance of the state.
The Unscented Transform (UT) is an excellent alternative to the linearization of the dynamics. It considers the state distribution as Gaussian and captures its mean and covariance through a set of sample points called *sigma points*.
The sigma points can then be propagated through the non-linear system, and, through certain weighted transforms the mean and covariance of the following iteration can be computed.

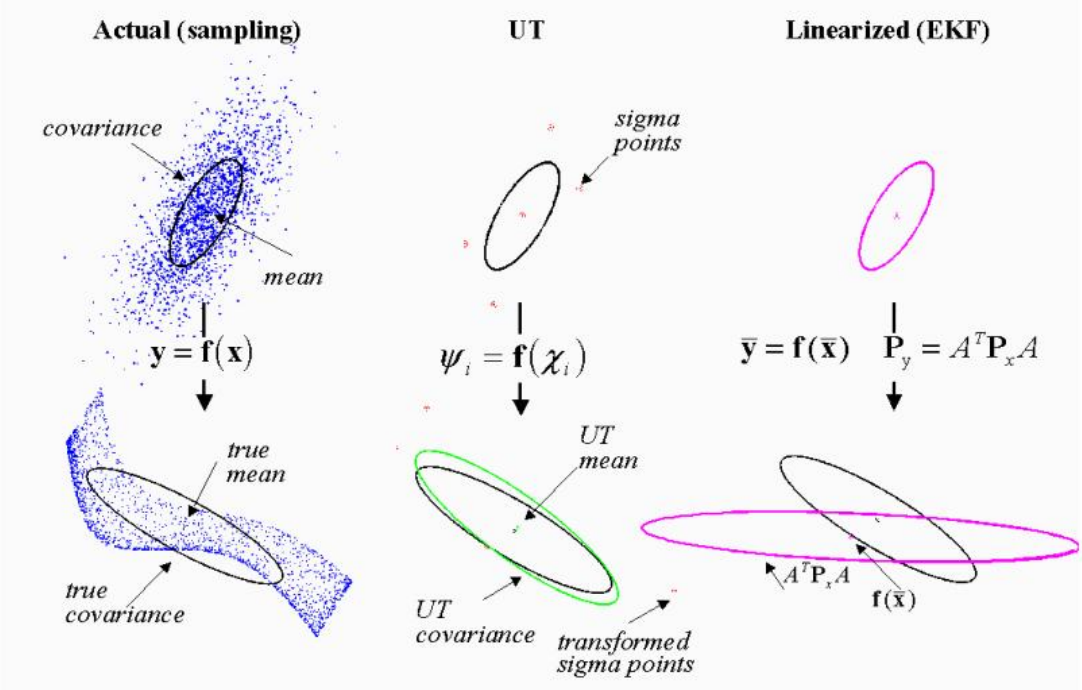


Figure 2.11: Comparison between the sampling procedure, the UT and the linearization

It can also be noted that, the UT shows greater performances with respect to a linearization when it comes to the estimation of the mean and covariance, as it approximates them to the second order instead of the first. This explains why the UKF can perform much better than the EKF when subjected to highly non-linear dynamics.
Finally, good estimates of the statistics of a process can be achieved with a relatively low computational burden as a limited amount of sigma points has to be considered. This differs significantly from sampling methods such as the Monte Carlo one, that needs a vast amount of points to obtain an accurate state distribution.
Considering a generic nonlinear system:

$$\begin{cases} \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \\ \mathbf{y}_k = h(\mathbf{x}_k, \mathbf{v}_k) \end{cases} \tag{2.31}$$

A set of sigma points can be defined from the state and the covariance matrix of the previous iteration.
This step is highly tunable, as the sigma points distribution depends on two coefficients (α, κ), which can be imposed by the user [WV00].
The sigma points are obtained as:

$$\hat{\mathbf{x}}_k^{(i)} = \hat{\mathbf{x}}_k^+ + \tilde{\mathbf{x}}^{(i)}, \qquad i = 1, ..., 2n \tag{2.32}$$

$$\tilde{\mathbf{x}}^{(i)} = \left(\sqrt{c\mathbf{P}_k^+}\right)_i^T, \qquad i = 1, ..., n \tag{2.33}$$

$$\tilde{\mathbf{x}}^{(n+i)} = -\left(\sqrt{c\mathbf{P}_k^+}\right)_i^T, \qquad i = 1, ..., n \tag{2.34}$$

Where:

$$c = \alpha^2(n + \kappa) \tag{2.35}$$

Sigma points are then individually propagated and then a weighted average is computed to obtain the predicted mean value of the state. Furthermore the predicted covariance matrix is obtained in a similar fashion.
This step also retains the dependence on the user defined parameters imposed previously, as in the weighted average the weight of each single sigma point depends on α, β and κ

$$\hat{\mathbf{x}}_{(k+1)}^{(i)} = f(\hat{\mathbf{x}}_k^{(i)}, \mathbf{u}_{k+1}) \tag{2.36}$$

$$\hat{\mathbf{x}}_{k+1}^- = \sum_{i=0}^{2n} W_M^{(i)} \hat{\mathbf{x}}_{k+1}^{(i)} \tag{2.37}$$

$$\mathbf{P}_{k+1}^- = \sum_{i=0}^{2n} W_c^{(i)} (\hat{\mathbf{x}}_{k+1}^{(i)} - \hat{\mathbf{x}}_{k+1}^-)(\hat{\mathbf{x}}_{k+1}^{(i)} - \hat{\mathbf{x}}_{k+1}^-)^T + \mathbf{Q}_k \tag{2.38}$$

Where:

$$\begin{aligned} W_M^{(0)} &= 1 - \frac{n}{c} & W_M^{(i)} &= \frac{1}{2c} \\ W_c^{(0)} &= (2 - \alpha^2 + \beta) - \frac{n}{c} & W_c^{(i)} &= \frac{1}{2c} \end{aligned} \tag{2.39}$$

Based on the predicted state, the corrected sigma points are defined as:

$$\hat{\mathbf{x}}_{k+1}^{(i)} = \hat{\mathbf{x}}_{k+1}^- + \tilde{\mathbf{x}}^{(i)}, \qquad i = 1, ..., 2n \tag{2.40}$$

$$\tilde{\mathbf{x}}^{(i)} = \left(\sqrt{c\mathbf{P}_{k+1}^-}\right)_i^T, \qquad i = 1, ..., n \tag{2.41}$$

$$\tilde{\mathbf{x}}^{(n+i)} = -\left(\sqrt{c\mathbf{P}_{k+1}^-}\right)_i^T, \qquad i = 1, ..., n \tag{2.42}$$

The prediction of the measurement is then computed using the corrected sigma points.

Firstly a propagation is performed using the measurement function for each single sigma state, afterwards the computation of its weighted average is performed and finally the covariance and cross covariance are obtained.
The process is performed as:

$$\hat{\mathbf{y}}_{k+1}^{(i)} = h(\hat{\mathbf{x}}_{k+1}^{(i)}) \tag{2.43}$$

$$\hat{\mathbf{y}}_{(k+1)}^{-} = \sum_{i=1}^{2n} W_M^{(i)} \hat{\mathbf{y}}_{k+1}^{(i)} \tag{2.44}$$

$$\mathbf{P}_y = \sum_{i=1}^{2n} W_c^{(i)} (\hat{\mathbf{y}}_{k+1}^{(i)} - \hat{\mathbf{y}}_{k+1}^{-})(\hat{\mathbf{y}}_{k+1}^{(i)} - \hat{\mathbf{y}}_{k+1}^{-})^T + \mathbf{R}_k \tag{2.45}$$

$$\mathbf{P}_{xy} = \sum_{i=1}^{2n} W_c^{(i)} (\hat{\mathbf{x}}_{k+1}^{(i)} - \hat{\mathbf{x}}_{k+1}^{-})(\hat{\mathbf{y}}_{k+1}^{(i)} - \hat{\mathbf{y}}_{k+1}^{-})^T \tag{2.46}$$

Finally, the updated estimate of the state can be obtained by the usual means of *Kalman gain*.

$$\mathbf{K}_{k+1} = \mathbf{P}_{xy} \mathbf{P}_y^{-1} \tag{2.47}$$

$$\hat{\mathbf{x}}_{k+1}^{+} = \hat{\mathbf{x}}_{k+1}^{-} + \mathbf{K}_{k+1}(\mathbf{y}_{k+1} - \hat{\mathbf{y}}_{k+1}^{-}) \tag{2.48}$$

$$\mathbf{P}_{k+1}^{+} = \mathbf{P}_{k+1}^{-} - \mathbf{K}_{k+1} \mathbf{P}_y \mathbf{K}_{k+1}^T \tag{2.49}$$

The UKF presents some solid advantages with respect to the other filtering technique, as it does not require the computation of Jacobians and can deal with highly non-linear situations.
However this advantages come with a great computational load as each sigma point needs to be propagated individually.

3 | Relative pose pipeline

3.1 Pipeline introduction

The aim of this dissertation is to create a relative navigation algorithm that leverages on a pre-existing NN pipeline. As such, both an image generation algorithm based on the SPEED database and a filtering architecture to improve the output had to be devised and implemented.

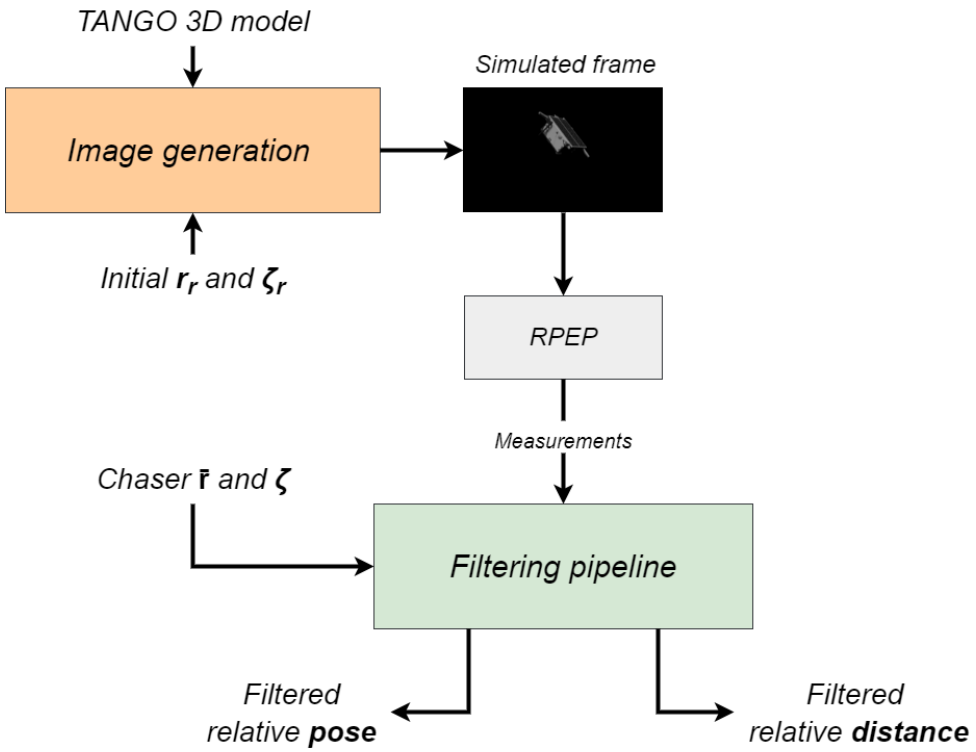


Figure 3.1: Overall architecture of the relative navigation pipeline

This chapter will initially tackle the image generation and then present the filtering pipeline.

3.2 Image generation procedure

The first step of the dissertation is the image generation step. Its main purpose is to generate: single frames, coherent image sequences and a replica of the SPEED database. This algorithm was essential as no image database, other than SPEED, was available and could fulfill our needs.

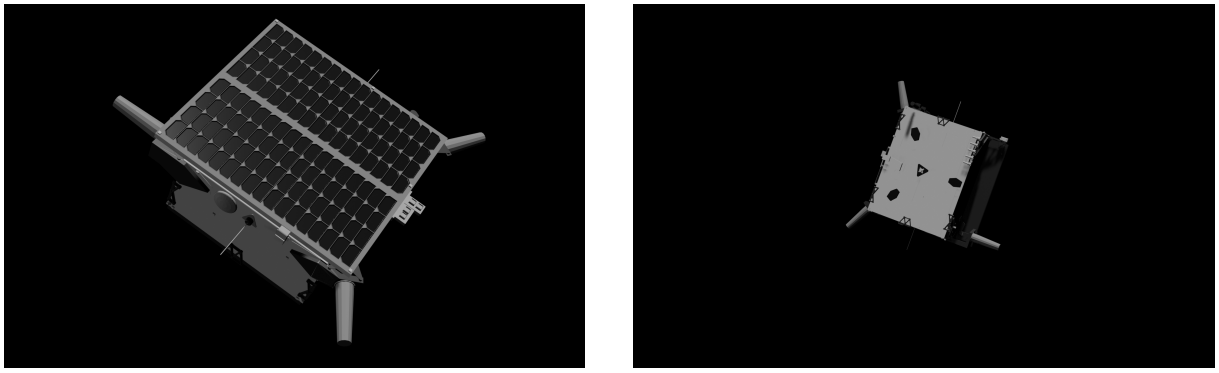


Figure 3.2: Example images generated from the pipeline

To build the pipeline the open source software *Blender* was chosen, as it was very appealing due to: its performance, its great online support and community and its scripting capabilities that allowed to fully control the scene via *Python*.
The overall product consists in an image generation environment capable of creating pseudo-realistic images of the TANGO spacecraft given as inputs the relative distance and attitude. Moreover, the pipeline is also capable of both propagating the dynamics and generating a coherent image sequence depicting the evolution of the relative motion. Salient features and challenges will be explained in the following sections.

3.2.1 Scene layout

First of all a layout for the scene had to be determined and the procedure for placing either the camera or the 3D model of TANGO had to be laid out.
The choice ultimately fell on placing the 3D model in the center of the scene and then generate the camera in the proper position. This decision was deemed convenient for many reasons. Firstly the model is made up of several components, each one with its own barycenter and coordinate system. This could pose a problem while performing a rigid body rotation as each piece may rotate around its own coordinate system leading to an un-coherent rotation of each single component.
Secondly, it was deemed easier to generate a camera in the correct position and orientation instead of moving and rotating around the whole 3D model.

Once a layout was chosen, it was necessary to identify the different reference frames at play, both in the attitude problem and in the *Blender* software.

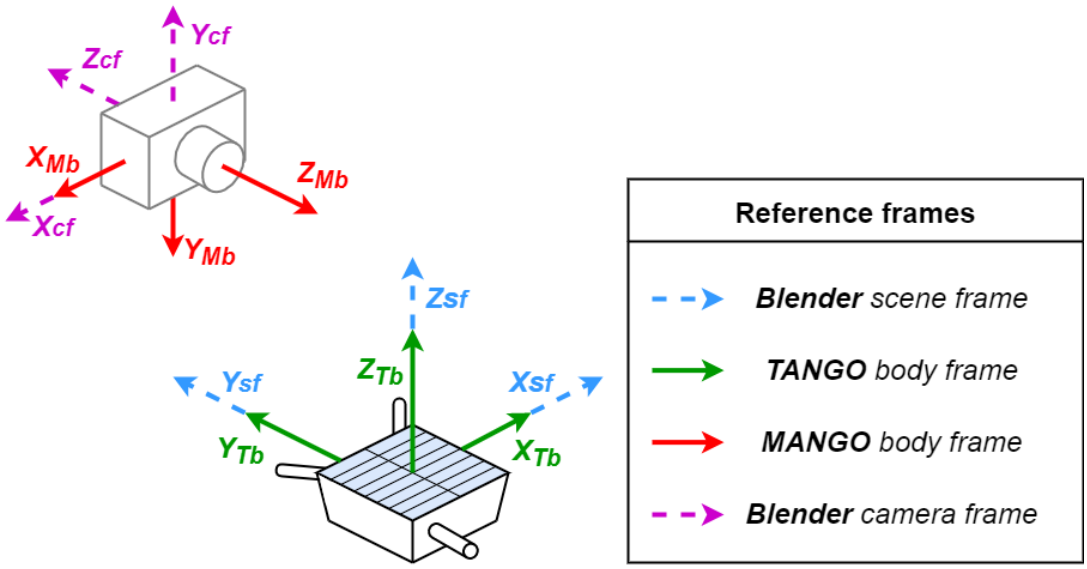


Figure 3.3: Scheme of the various coordinate systems in the scene

As can be seen from Figure 3.3, four main reference frames are involved, although for convenience they can be reduced to three as TANGO body frame was imposed coincident to *Blender* scene frame. Furthermore, a key assumption was considered as MANGO body frame was set coincident to the camera frame defined by the sensor on-board. This was deemed reasonable and its correctness was further proved by testing this assumption. The results can be found in Section 4.1.1.
Note that, for the remainder of this dissertation, the terms *MANGO body frame* and *Chaser camera frame* will be used to indicate the same reference frame.

Once the reference frames were identified, the relations between each of them had to be laid out since they are of paramount importance to properly generate the virtual camera. Focusing on the relation between *Blender* camera frame and the chaser camera frame: the software uses a completely different notation, as it imposes the Z axis opposite of the boresight of the camera. Fortunately the relation between the two frames is quite simple as a rotation of $[\pi$ rad$]$ about their common X axis is enough to realign them.
Meanwhile, the relation between TANGO and MANGO body frames is more complex and is related to the dynamics explained in Section 2.1.3 through the rotation matrix $\mathbf{\Gamma}$.

Moving on to camera placement, it is important to recall that the relative separation is expressed in the chaser body frame. Meanwhile, in the *Blender* pipeline, the target body frame is imposed as the fixed one, therefore the relative distance has to be expressed in

that reference frame. This is done as:

$$\boldsymbol{r}_{r_{Tb}} = -(\boldsymbol{\Gamma}^T \boldsymbol{r}_{r_{Mb}}) \tag{3.1}$$

Where $\boldsymbol{r}_{r_{Tb}}$ is the inverted relative distance vector expressed in the TANGO body frame and indicates the position at which the camera must be placed. Meanwhile, $\boldsymbol{r}_{r_{Mb}}$ is the relative distance vector expressed in the MANGO body frame.

Once the relative distance has been evaluated, the actual camera can be positioned in the obtained point. Its attitude is then adjusted to point to the target, in the following manner:

$$\boldsymbol{R}_{camera} = \boldsymbol{R}_\pi \boldsymbol{\Gamma} \qquad where \qquad \boldsymbol{R}_\pi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{3.2}$$

Where $\boldsymbol{R}_{camera}$ is the attitude to be imposed to the camera to correctly point to the target and $\boldsymbol{R}_\pi$ is the π rad rotation needed to correct the boresight of the camera.

3.2.2 Lighting conditions

To allow the creation of realistic images of the TANGO spacecraft special care had to be put into the illumination conditions.

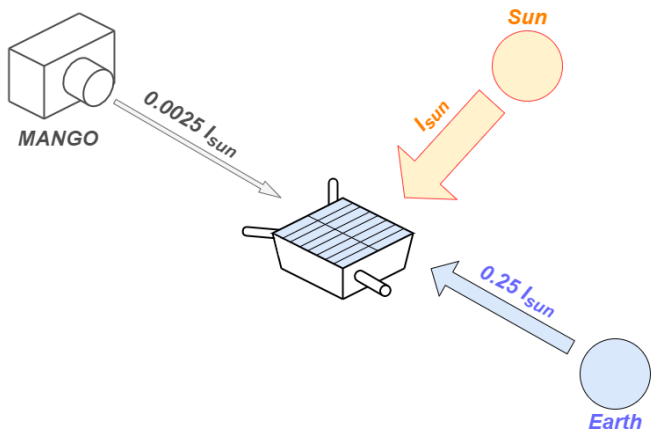


Figure 3.4: Light sources scheme

As can be seen from Figure 3.4, three light sources have been implemented: the main two being the Sun and Earth, along with a third dimmer light placed in correspondence of the chaser. Its main purpose is to reduce the sharp contrast between illuminated and dark regions.

Focusing now on the Sun, its position has been obtained from the spacecraft orbit and from the considered season as:

$$\begin{gathered} S_n = [\cos(\omega_{\oplus} t), \quad \sin(\omega_{\oplus} t)\cos(\epsilon_{\oplus}), \quad \sin(\omega_{\oplus} t)\sin(\epsilon_{\oplus})] \\ S_c = \boldsymbol{A}_{B/N} S_n \qquad and \qquad S_t = \boldsymbol{\Gamma}^T S_c \end{gathered} \tag{3.3}$$

Where S_n is the Sun versor in the Earth Centered Inertial (ECI) frame, S_c is the Sun versor in the chaser body frame and S_t is instead expressed in the target body frame. $\omega_{\oplus}$ represents the Earth angular velocity about the Sun and $\epsilon_{\oplus}$ represents its inclination with respect to the Ecliptic plane. $\boldsymbol{A}_{B/N}$ represents the rotation between the chaser body frame and the ECI frame, which is found by propagating the chaser dynamics.
Note that, for simplicity, the spacecraft has been considered coincident to Earth when dealing with its seasonal illumination and Earth orbit has been considered circular.
As such, frames present both seasonal and orbital variations in their illumination conditions.

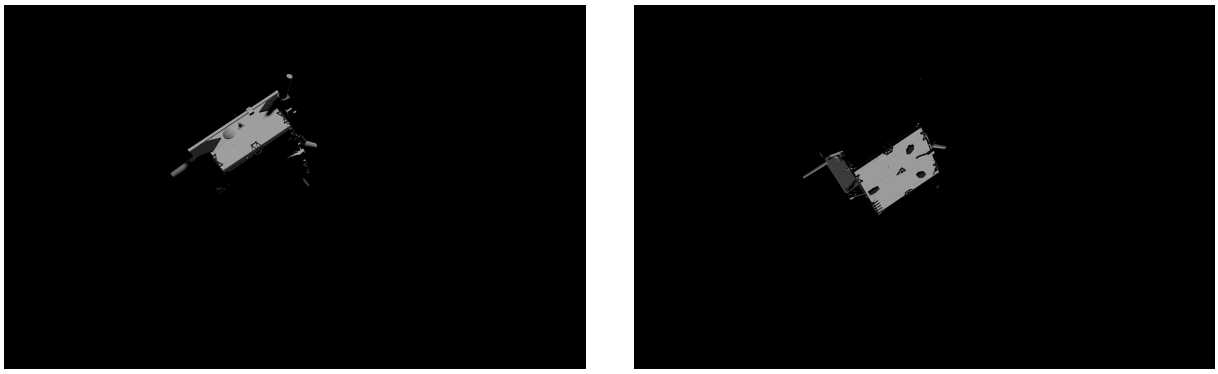
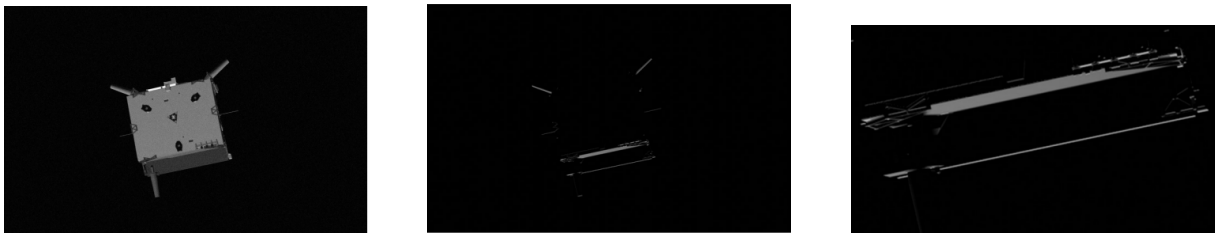


Figure 3.5: Example of seasonal variations in the lighting condition (Note: the effect has been exaggerated for better visibility)

Considering now Earth albedo, its direction was easier to implement as the relative position between the spacecraft and Earth is fully available from the dynamics considered in subsection 2.1.2. For what concerns the related illumination, it was chosen to consider it as 25% that of the Sun, which is roughly the amount of energy coming from the Sun and reflected by the Earth.
At last, considering the smaller intensity light positioned on the chaser, although not physically accurate it was deemed necessary to reduce the sharp contrast between light and shadow in the frames.

As can be seen from Figure 3.6, the SLN encountered difficulty when trying to recognize the spacecraft in such illumination conditions.



(a) SPEED image (b) Simulated image (c) Detected BB

Figure 3.6: Example of a failed detection due to bad lighting (No fill light present)

An alternative approach to the added light could have been a volumetric simulation of the atmospheric scattering, but, due to the excessive computational load and rendering time it was chosen to go for the simpler and more direct route.

3.2.3 Texturing

The implementation of realistic textures on the spacecraft model was an important part of the overall pipeline.
Unfortunately, the 3D model provided by OHB-Sweden [Swe] did not present any texture or data on the materials, thus their property was not available. It was decided to proceed by iteratively adjusting the textures on the spacecraft and comparing them to the SPEED database over several images.

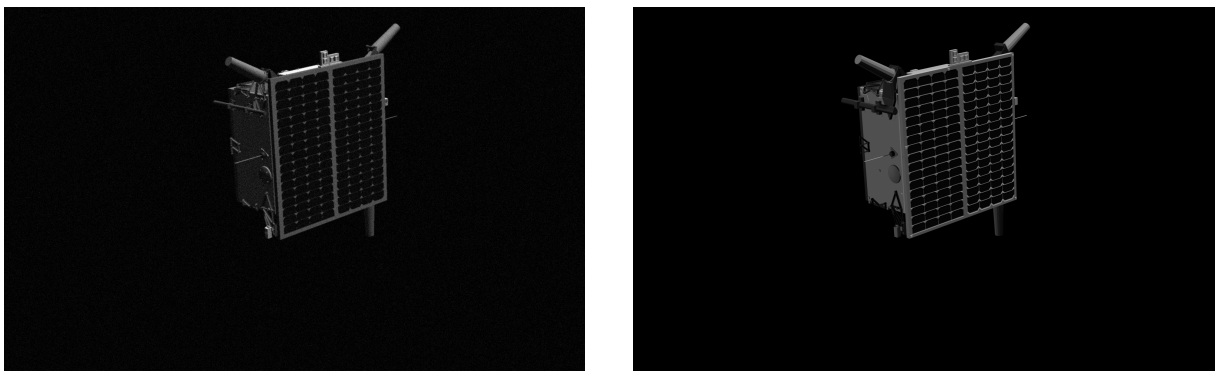


Figure 3.7: Comparison between the SPEED database [left] texturing and the replica [right] texturing

Although a perfect match in the texture was not achieved, it was deemed accurate enough for the purpose of the dissertation.
Unfortunately, the presented mismatch led to a minor loss of performance of the pipeline, though it was partly counteracted by the decision to avoid Earth as a background.

This approach was also adopted to avoid problems regarding both the scale of the scene and potential artifacts in the lighting. In addition, it also implied a big reduction of the rendering times.
As can be seen from the results in Section 4.1.1 this approach was successful as no major discrepancies were found between the SPEED database and the replica generated by our pipeline.

3.2.4 SPEED comparison

As mentioned in the previous subsection, the comparison with the SPEED database was performed to further validate the image generation approach above-mentioned.
To enable this, the recreation of all the images present in the SPEED database was pursued. Some examples are proposed in Figure 3.8, note that the the lighting condition is not matching as SPEED did not provide any information about it.

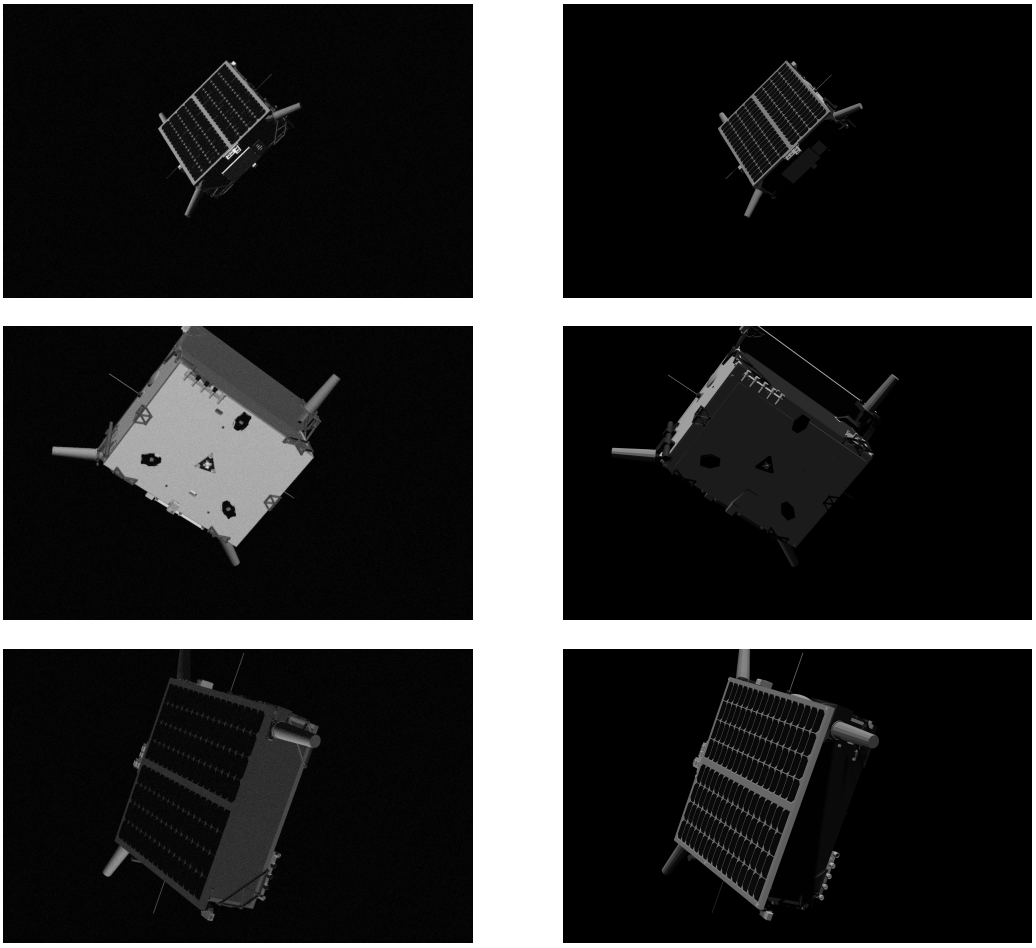


Figure 3.8: Comparison between the SPEED database [left] and the replica database [right]

Once the mockup database was available, to ensure its correctness some steps were taken: firstly a preliminary visual comparison between the two databases was performed with the aim of detecting evident errors through the use of superposition techniques.
Afterwards the whole dataset was ran through the RPEP and its results were then compared to the ones obtained from the SPEED database. The results of such analysis are found in Section 4.1.1.

3.3 Sequence generation

To finalize the presented image generation pipeline the creation of coherent image sequences had to be implemented.
Said algorithm enables the autonomous rendering of a large number of frames. Those can be both a collection of single uncorrelated images (e.g. SPEED) or full sequences of coherent frames based on the relative dynamics propagation.
In the following sections a more in depth analysis will be performed.

3.3.1 Propagation procedure

The propagation step is based on the dynamics equations provided in Section 2.1.
To properly enable the procedure, a set of initial conditions must be identified and imposed. Those are: initial relative separation and attitude, orbital parameters of the chaser and initial angular velocities of both target and chaser.

Starting from the latter, the angular velocity of the chaser has been imposed in a way that allows the spacecraft body frame to track the LVLH frame.
Instead the target angular velocity is generated from a distribution obtained from a preliminary study of the light curves of tumbling LEO objects [Uni21]. Although this approach is quite accurate and realistic, it presents a couple of drawbacks: firstly the distribution presents a very big population of objects with very high tumbling rates, below $5s$, which can pose problems to the filtering stage and therefore needs to be accounted.
Secondly there is an evident bias towards faster tumbling rates. This is due to the presence of objects with multiple reflecting surfaces, which generate more peaks on their light-curves per revolution and hence will be interpreted as a faster tumbling rate. This bias cannot be counteracted or accurately estimated, as the geometry of the majority of the tracked objects is unknown, and thus must be accepted.

Moving on to the imposed orbital parameters, a circular, equatorial, with semimajor-axis equal to *10.000 km* has been chosen as it proved to be a simple and manageable case

study.

Finally, regarding the relative quantities, the initial conditions have been extracted from SPEED ground truths. As such, the initial frame of each sequence can be traced back to a frame from the database. Note that, by following this approach, the recreation of the full SPEED collection can be easily performed by saving only the first frame of the procedure.

In addition to the initial condition, a set of tuning parameters have to be imposed by the user.
Initially the propagation time of the relative dynamics has to be properly enforced.
Afterwards the Frames Per Second (fps) must be defined: it represents the number of images rendered for each second of the propagation. This dissertation will mostly use an acquisition rate of 2 fps (one render taken every 0.5 s) as it seemed a reasonable and conservative value for spaceborne applications.

3.3.2 Blender pipeline layout

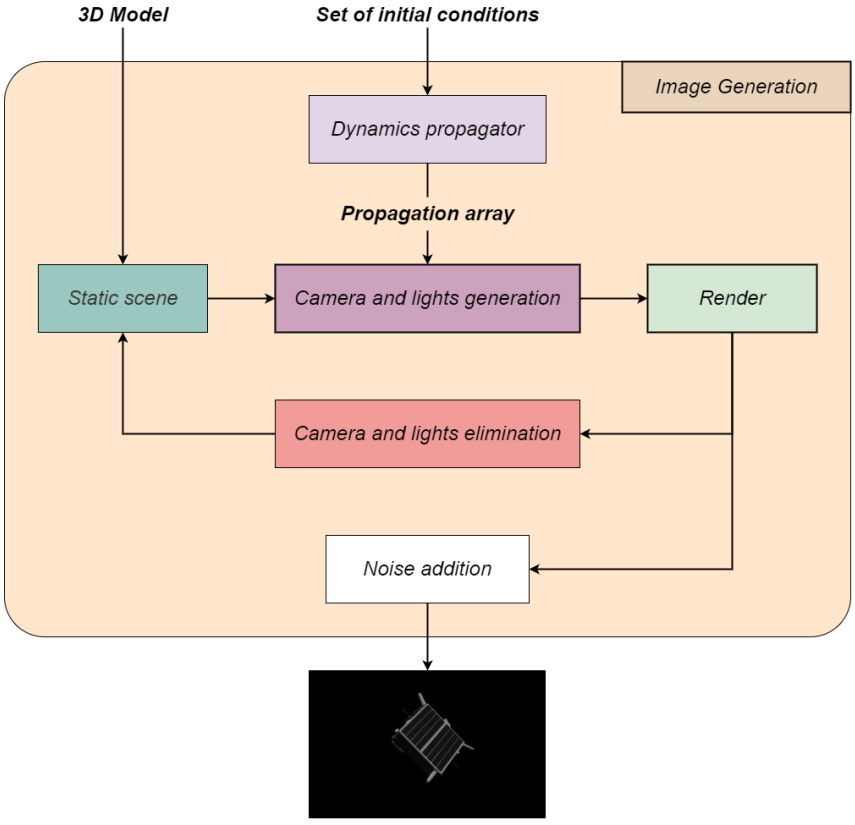


Figure 3.9: Scheme focusing on the image generation procedure

The outline of the Image generation step of the pipeline is presented in detail in Figure 3.9.
First of all, a set of initial conditions is fed to the pipeline. Those are processed through a numerical ODE propagator that estimates the evolution of the state in the considered timeframe. The desired fps are then taken into account by downsampling the propagation accordingly in order to obtain the state of each desired frame.

The aforementioned array is then progressively fed through the actual rendering cycle.
At first the scene is defined as static and presents only the 3D model of TANGO in its center. Once the state is fed through the cycle the next step can begin. The corresponding camera and its matching set of lights is placed in the scene and a rendering operation is performed.
Note that Blender presents two different rendering engines named EEVEE and Cycles. While the latter is more powerful the first is sensibly faster. As such, this dissertation will make use of EEVEE as a huge number of frames was needed in a relatively short timespan.
Once the rendering operation has been performed the camera and the set of lights are permanently removed from the scene, effectively leaving it in its static condition and ready for a new iteration.

Focusing instead on the obtained image, initially the clean frame is properly named and saved, afterwards a random gaussian noise filter with covariance *0.0022* [Kis+20] is applied and the noisy frame is then saved.

3.4 Filtering pipeline

To improve the performance of the pipeline a filtering step was deemed necessary. to this aim a set of Kalman filters was employed as the tackled problem has sequential properties.

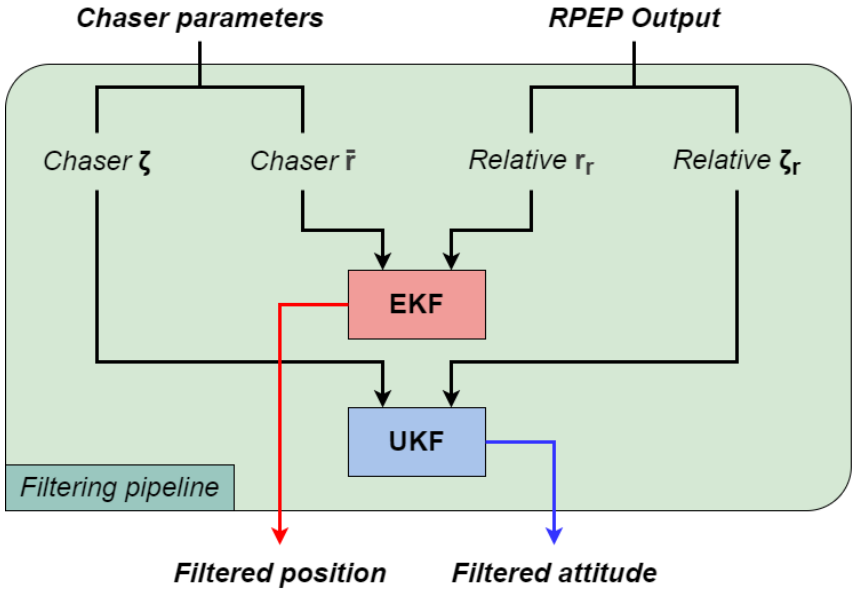


Figure 3.10: Scheme focusing on the filtering step of the pipeline

Image 3.10 describes the layout of the filtering procedure implemented in this dissertation. As can be seen both an EKF and a UKF have been used for the two different dynamic conditions. To allow the decoupling of the relative attitude problem from the relative distance problem one key assumption had to be put in place: since the correlation between the two problems lies in the chaser inertial attitude and position, it has been considered fully known and available, thus allowing the uncoupling of the dynamics.
This was deemed a reasonable assumption given that a chaser spacecraft is commonly equipped with high accuracy attitude sensors and has full ground support. Thus its inertial attitude and position can be assumed fully known with a high degree of accuracy.

The EKF was preferred when dealing with the relative position dynamics as the problem presents quite linear characteristics and a slower evolution. Moreover, as the linearization is quite effective, the filter showed excellent performances and a reduced computational load.

Whereas, regarding the relative attitude problem, due to its faster evolution and highly non-linear behaviour an approach involving the UKF was pursued. Although the computational time dedicated to this step is higher, the filter proved to be quite stable and performing.

3.4.1 EKF specifics

As previously mentioned the EKF was chosen to tackle the relative dynamics problem presented in Section 2.1.2.
Those equations have been directly implemented in the filter and propagated during the prediction step using an ODE solver. Moreover, the *Jacobian* matrix of both the state propagation equation and measurement equation have to be computed to fully implement the filter. This is done as:

$$\mathbf{F} = \frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{\partial \ddot{x}}{\partial x} & \frac{\partial \ddot{x}}{\partial y} & \frac{\partial \ddot{x}}{\partial z} & 0 & \frac{\partial \ddot{x}}{\partial \dot{y}} & 0 \\ \frac{\partial \ddot{y}}{\partial x} & \frac{\partial \ddot{y}}{\partial y} & \frac{\partial \ddot{y}}{\partial z} & \frac{\partial \ddot{y}}{\partial \dot{x}} & 0 & 0 \\ \frac{\partial \ddot{z}}{\partial x} & \frac{\partial \ddot{z}}{\partial y} & \frac{\partial \ddot{z}}{\partial z} & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{H} = \frac{\partial h}{\partial \mathbf{x}} = \begin{bmatrix} & & & 0 & 0 & 0 \\ & \boldsymbol{A}_{B/L} & & 0 & 0 & 0 \\ & & & 0 & 0 & 0 \end{bmatrix} \tag{3.4}$$

The fully developed derivatives of the Jacobian Matrix $\mathbf{F}$ are available in Appendix A
To finalize the setup of the filter also the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ had to be imposed, where $\boldsymbol{Q}$ represents the process noise and $\boldsymbol{R}$ is the measurement noise.
As the process is considered perfectly known and deterministic, the corresponding matrix $\boldsymbol{Q}$ is considered null. Instead, the measurement noise was carefully studied and characterized analyzing the results from the RPEP on the simulated image database. The full process can be found in Section 4.1.1

3.4.2 UKF specifics

Moving on to the relative attitude problem, as mentioned previously, an UKF has been deemed more suitable.
To characterize the filter no Jacobian matrices were needed, instead, the spread and the weight of the various *sigma points* need to be characterized. To this avail three parameters have been introduced:

- α, identifies their spread around the mean value.
- κ, represents a second scaling factor.
- β, affects the weighted average of the sigma points.

The three parameters (α, κ, β) have been respectively imposed as: $[10^{-3}, 0, 2]$ following suggestions from the literature [WV00] and a brief trial and error testing phase.

Analogously to the EKF the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ had to be imposed.
Regarding the process noise, a similar approach has been pursued, as such it was imposed null.
Conversely, an accurate study of the measurement covariance has been performed taking into account various factors. A full analysis is present in Section 4.3.2

Moreover, during the analysis of the most common failure modes, a scarce robustness to outliers has been detected. As this problem led to various localized failures of the filter, countermeasures were put in place.
It was chosen to follow an approach based on the *Mahalanobis distance* between expected and real measurements. If the aforementioned parameter is greater than the *2 sigma* threshold (obtained from the inverse χ^2 distribution), then its corresponding measurement is discarded and the update phase is completely skipped, meaning that the probability of measurement happening under the current state estimate is less than 4.6%.

3.4.3 UKF using MRPs

In this dissertation MRPs are consistently used throughout the filtering sequence.
The main driver for this choice was that the dynamics has been expressed in MRPs, so the state had to follow that logic. Therefore, on one hand Euler angles were avoided in the UKF due to the non-linear relation between them and the state expressed in MRPs and the added complexity that it implied [SJ95].
On the other hand, quaternions needed a complex correction step in order to eliminate the use of additive errors [LMS82] due to the norm constraint, as such it was chosen to avoid them.

Unfortunately the use of MRPs brought forth a number of problems.
Firstly a way to account for the switching between the shadow and the regular set had to be devised: whenever the norm of the MRP describing either the state or the measurements was greater than one the switching procedure was performed.
To this avail the equation mapping the regular MRP set to its shadow counterpart was considered as [KS09]:

$$\zeta_S = -\frac{\zeta}{\zeta^T\zeta} \tag{3.5}$$

Furthermore, a similar transformation had to be applied to the corresponding covariance matrix as:

$$\boldsymbol{P}_k^S = \boldsymbol{\Lambda}\boldsymbol{P}_k\boldsymbol{\Lambda}^T \qquad \text{where} \qquad \boldsymbol{\Lambda} = 2\zeta^{-4}(\zeta\zeta^T) - \zeta^{-2}\boldsymbol{I} \tag{3.6}$$

Where $\boldsymbol{P}_k^S$ is the covariance matrix expressed in the shadow set.

Although this approach seemed to improve the performance of the filter, it generated the possibility of mismatches in the switching procedure. For example, at a certain iteration the measurement could be switched to shadow, but not the state, since it could be slightly below the threshold. In that case a mismatch would happen and spike in the error graph would be detected. This error spike would be a "false negative", considering that both the shadow MRP and its regular form indicate roughly the same attitude. As such a procedure to correct this *false errors* must be devised.
Finally, a more in depth analysis of the measurement noise covariance matrix ($\boldsymbol{R}$) was needed, as correlations between it and the MRP norm were deemed possible. The overall analysis will be found in Section 4.3.2.

4 | Results

4.1 RPEP testing

The RPEP played a crucial role during the testing phase of this dissertation, as its capabilities were used for both the measurement extraction and the validation of the simulated frames.
Overall the pipeline showed very good performance, managing a centimeter and degree level accuracy on both the SPEED database and its replica.
The obtained results will be now presented.

4.1.1 Replica and real SPEED comparison

As the database generated from the image generation pipeline presented some evident mismatches with the real SPEED database, a way to compare the two had to be devised. It was chosen to exploit the RPEP pipeline to quantify its performance loss, which was assumed proportional to the difference between the databases.
The RPEP performance on both image collections are presented in Tables 4.1 and 4.2:

Absolute error		
	Mean	***Median***
E_t	*10.36 cm*	*3.58 cm*
$\boldsymbol{E_t}$	*[0.52, 0.56, 10.25] cm*	*[0.24, 0.27, 3.50] cm*
E_θ	*2.24°*	*0.81°*
$\boldsymbol{E_\theta}$	*[1.57°, 0.84°, 1.72°]*	*[0.52°, 0.33°, 0.34°]*
Standard Deviation		
$\boldsymbol{\sigma_{E_t}}$	*[1.62, 1.71, 30.44] cm*	
$\boldsymbol{\sigma_{E_\theta}}$	*[8.92°, 5.11°, 10.82°]*	

Table 4.1: RPEP performance on the SPEED database

Absolute error		
	Mean	***Median***
E_t	*10.10 cm*	*3.52 cm*
$\boldsymbol{E_t}$	*[0.58, 0.65, 9.96] cm*	*[0.33, 0.32, 3.41] cm*
E_θ	*2.32°*	*0.94°*
$\boldsymbol{E_\theta}$	*[1.65°, 0.83°, 1.65°]*	*[0.56°, 0.37°, 0.46°]*
Standard Deviation		
$\boldsymbol{\sigma_{E_t}}$	*[1.79, 2.45, 39.82] cm*	
$\boldsymbol{\sigma_{E_\theta}}$	*[9.41°, 4.72°, 9.91°]*	

Table 4.2: RPEP performance on the replica database

As can be seen the results between the two databases are very similar presenting only minor differences. As such, the use of our simulated frames has been deemed acceptable for the purpose of this dissertation.

Going more in depth in the result analysis, it is evident that the measurements of the relative distance on the Z axis are subjected to higher errors. This can be traced back to the BB-based correction of the outliers performed by the RPEP which presents the higher uncertainty on the boresight direction, thus providing a poorer result on the Z camera axis.

Moving on to the relative attitude, an uneven spread of the errors have been detected. It was proposed that this unevenness has to do with the position of the selected keypoints, as small errors on the detection of the features farther away from the center have a stronger effect than those closer.
This, in turn, creates biases on the axis characterized by the presence of such keypoints.

Further analysis was performed to identify possible biases on the simulated database, as the placement of the 3D model in the *Blender* environment was performed manually. Thus, it could introduce small biases in the overall database due to poor placement.
To this aim an initial study of the error distribution graphs was performed.

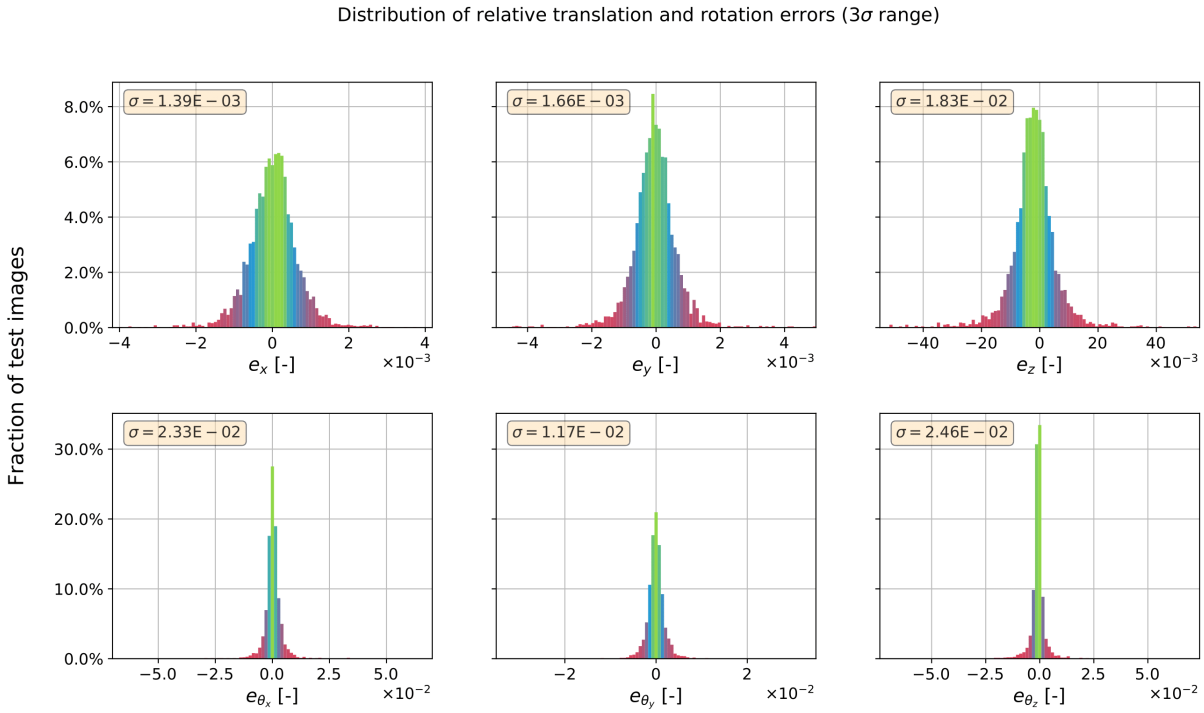


Figure 4.1: Relative error distribution on the simulated database

As can be seen from the graphs in Figure 4.1 no major bias is present as all the gaussian curves seem to be centered on a zero mean. However, a more in depth analysis highlights a slight bias as can be seen in Table 4.3:

Simulated database mean error	
	Mean error
Relative distance	*[0.01803, 0.02821, 1.764] cm*
Relative attitude	*[0.03520, 0.003274, 0.1062] deg*

Table 4.3: Mean error of the replica database

Overall the bias regarding the relative distance is quite contained being well below the centimeter level except for the boresight direction.
Similarly, the relative attitude error presents a very contained bias, below the degree level. In the end, both biases will be considered acceptable for the aim of this dissertation, as they are quite limited and generally fall one order of magnitude below the mean of the absolute error.

Overall the analysis of the results of the RPEP showed how, although not perfect, the proposed image generation procedure can provide an available and valuable alternative

to the SPEED database.

4.1.2 Results from image sequence

After the validation of the image generation procedure proposed in this dissertation, the analysis of complete and coherent image sequences could be attempted.

To this aim a few image sequences were first generated and evaluated by the RPEP, then filtered and finally the corresponding results were thoroughly investigated. A selection of two of those will be proposed in this section.

The first test case presents the *TANGO* spacecraft placed at a relative distance of around *12m* with a relatively fast rotational period of around *10s*. Its initial relative parameters were taken from *"Img000040"* of the SPEED database.
The first 12 frames of the sequence are visualized in Figure 4.2:

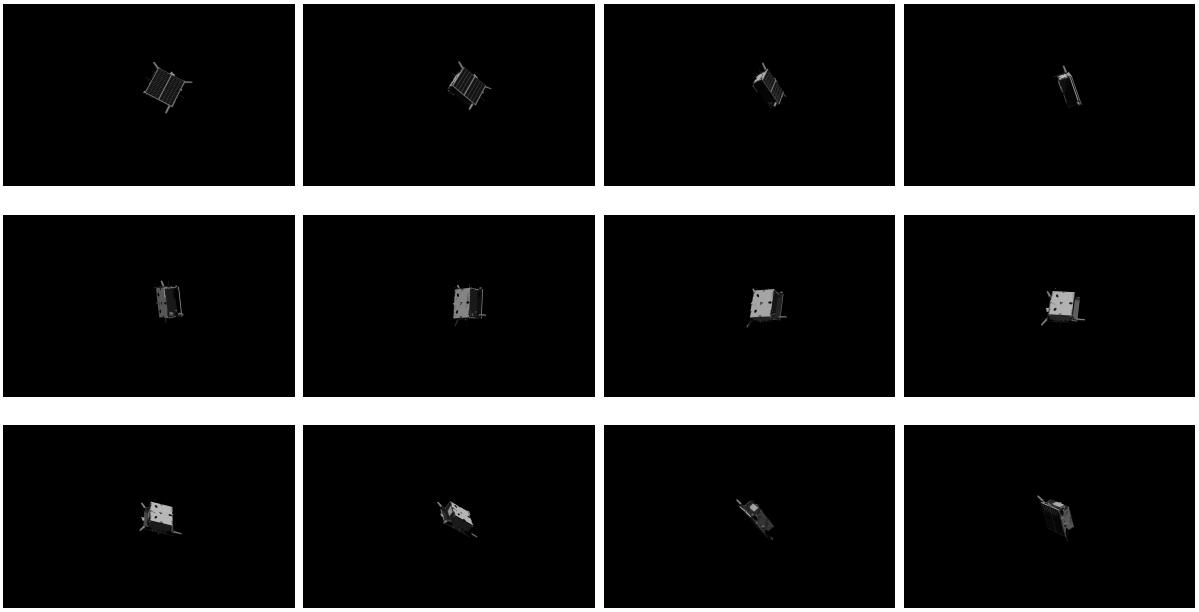


Figure 4.2: First 12 frames of the simulated sequence based on *"Img000040"* of the SPEED database

The generated frames were then passed through the RPEP on the *Colab* platform that used a high-end GPU to perform the inference. The results of the analysis can be seen in Table 4.4:

Absolute error		
	Mean	***Median***
E_t	*8.36 cm*	*5.73 cm*
$\boldsymbol{E_t}$	*[0.50, 0.48, 8.26] cm*	*[0.37, 0.30, 5.67] cm*
E_θ	*2.74°*	*1.13°*
$\boldsymbol{E_\theta}$	*[1.47°, 0.69°, 2.47°]*	*[0.75°, 0.46°, 0.52°]*
Standard Deviation		
$\boldsymbol{\sigma_{E_t}}$	*[0.70, 1.22, 15.69] cm*	
$\boldsymbol{\sigma_{E_\theta}}$	*[5.22°, 1.30°, 14.44°]*	

Table 4.4: Results obtained from the RPEP on the first sequence

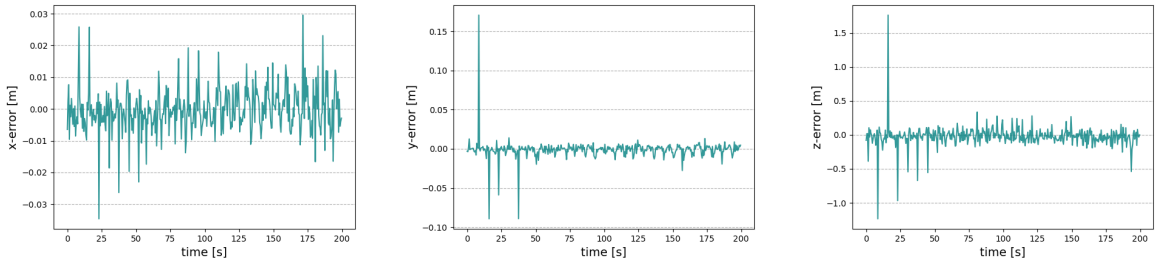


Figure 4.3: Error graphs of the relative separation components obtained from the *"Img000040"* sequence

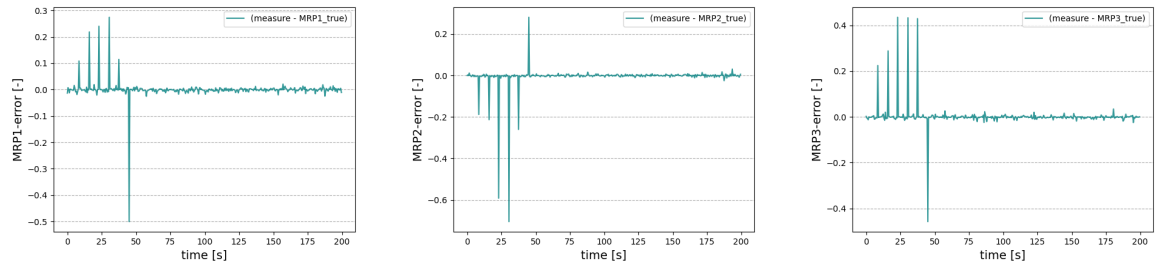


Figure 4.4: Error graphs of the MRPs obtained from the *"Img000040"* sequence

Overall the pipeline performance was better than expected presenting relatively low mean errors compared to the results found in Section 4.1.1. It was assumed that this was correlated to the relatively low distance taken into consideration, as the RPEP performance heavily depends on it.

Unfortunately, a few outliers were also detected both within the relative separation and the relative MRPs, but they will provide a good basis for the validation of the filtering sections.

The second test sequence considered retained roughly the same relative distance but had a much slower rotation period of around *2 min*. Its initial relative parameters were taken from *"Img000051"* of the SPEED database. The first 12 frames of the sequence are visualized in Figure 4.5:

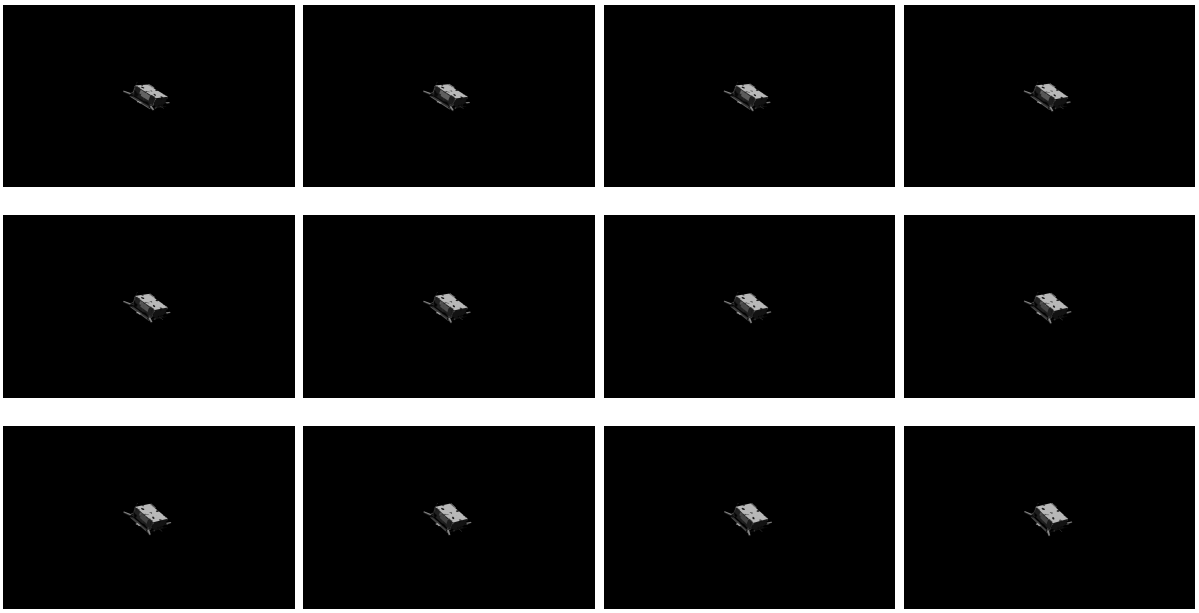


Figure 4.5: First 12 frames of the simulated sequence based on *"Img000051"* of the SPEED database

The results obtained from the sequence can be seen in Table 4.5:

Absolute error		
	Mean	***Median***
E_t	*14.41 cm*	*6.51 cm*
$\boldsymbol{E_t}$	*[0.64, 0.67, 14.30] cm*	*[0.42, 0.40, 6.50] cm*
E_θ	*4.94°*	*1.23*
$\boldsymbol{E_\theta}$	*[3.84°, 0.95°, 4.65°]*	*[0.80°, 0.62°, 0.47°]*
Standard Deviation		
$\boldsymbol{\sigma_{E_t}}$	*[1.10, 1.46, 43.10] cm*	
$\boldsymbol{\sigma_{E_\theta}}$	*[14.52°, 1.78°, 19.88°]*	

Table 4.5: Results obtained from the RPEP on the second sequence

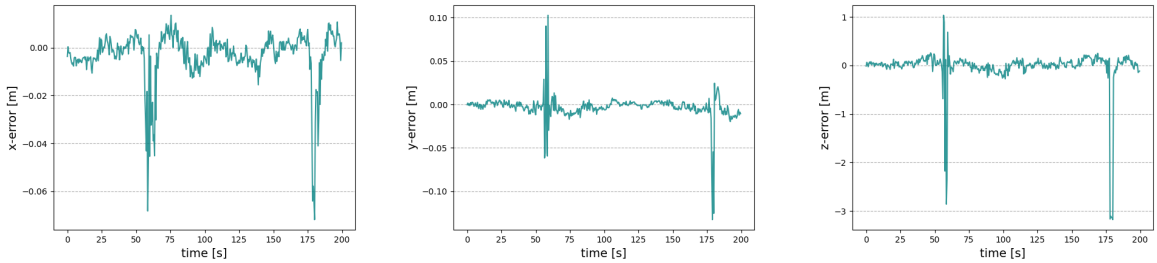


Figure 4.6: Error graphs of the relative separation components obtained from the *"Img000051"* sequence

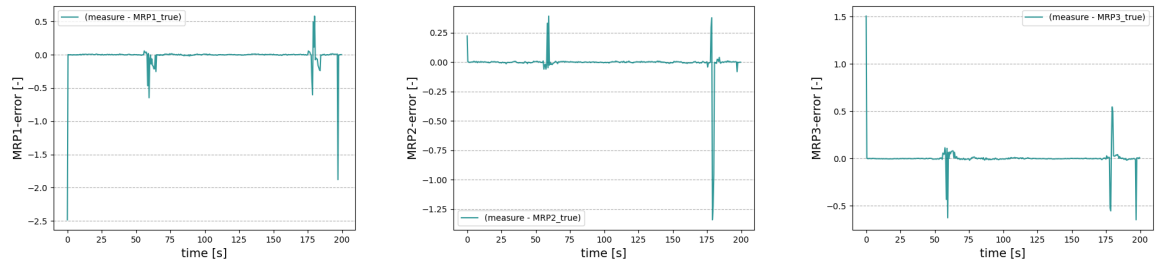


Figure 4.7: Error graphs of the MRPs obtained from the *"Img000051"* sequence

Overall the pipeline showed poorer performance with respect to the sequence analyzed before. Infact the mean error was higher but the median was quite similar: this phenomenon is a strong indicator of a higher presence of outliers within the results.
Considering the graphs of Figures 4.7 and 4.6 it is quite evident that outliers are indeed present but they tend to concentrate on certain regions of the sequence.
It has been speculated that this has to do with the slower rotational period which implies a small difference between successive frames. As such, if an outlier is detected in a particular pose condition, the following frame is very likely to be an outlier too as the spacecraft pose did not change by much.

Overall the pipeline performed with decent accuracy and a reduced number of outliers on the analyzed sequences, although a further increase of the quality of the results and an elimination of the outliers is to be expected in the following filtering procedure.

4.2 Extended Kalman Filter

Due to the paramount importance of the EKF to the performance of the whole filtering pipeline, its robustness had to be ensured.
Overall the EKF was tested both on real inputs obtained from the RPEP and on fully simulated inputs obtained by applying proper noise to the GT vector.

Altogether, the EKF was tested thousands of times and showed a very good robustness and perfect reliability.

4.2.1 Actual image sequence results

To make sure the EKF was robust to real inputs a testing campaign was performed on real image sequences. To the aim of this dissertation the sequences proposed in Section 4.1.1 will be analyzed.

Starting from the first proposed sequence, its results were fed to the EKF and an initial analysis was performed.
To facilitate the analysis of both the filtered and raw measurements a superimposition of the two was performed in the graphs. The scale has also been set to logarithmic to further improve legibility. The visualized graphs are presented in Figure 4.8:

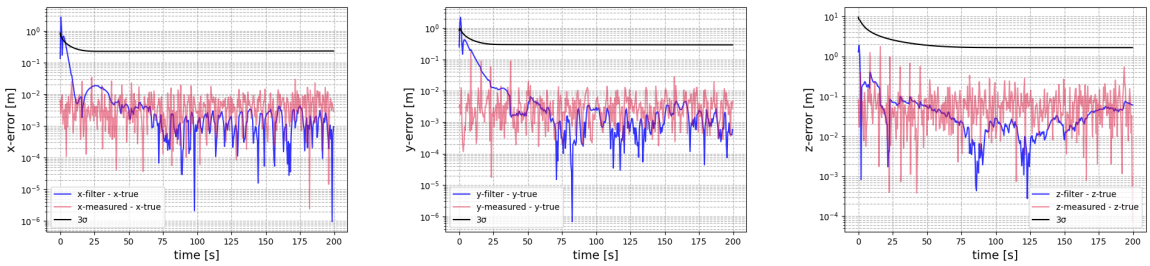


Figure 4.8: Filtered relative separation component graphs of the *"Img000040"* sequence

As can be clearly seen from Figure 4.8, the EKF manages to successfully filter out any outliers and reduce the error of up to one order of magnitude.
It can also be observed that the filter manages to settle around a millimeter level accuracy on the X and Y axis within the first few seconds, a clear improvement on the measurements obtained from the RPEP.
Unfortunately, the results associated to the Z axis are not quite as accurate managing only to reach a centimeter level accuracy. This was expected and can be justified by the poorer quality of the inputs, as they are generally associated to a higher error, as can be seen from Table 4.4.

Moving on to the second sequence, it presented a lower accuracy input and more concentrated outliers, as can be seen by looking at the results of Table 4.5. The results obtained from the EKF are reported in Figure 4.9:

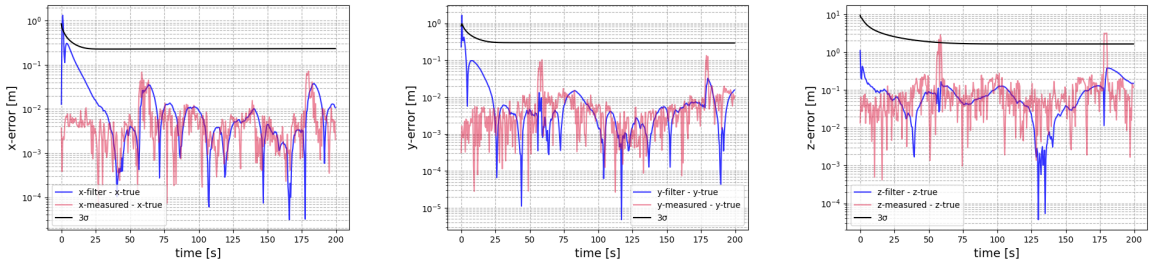


Figure 4.9: Filtered relative separation component graphs of the *"Img000051"* sequence

Looking at the results the robustness of the EKF is quite evident as, despite the inaccurate input, it managed to quickly converge to low error levels, comparable to the previous test case.
Unfortunately a strong decrease in the performance and an increase in the error can be observed in proximity of the outliers. This is caused by the absence of an outlier detection and scrapping procedure, meaning that every input is used in the update step of the EKF. Nonetheless, the filter proved to be very performing and its implementation was not deemed necessary for the purpose of this thesis.

In the end, the pipeline behaved properly when subjected to real inputs, proving its high robustness and absolute reliability even in the face of concentrated outliers. To further prove it an extensive analysis on a vast amount of simulated test cases have been performed.

4.2.2 Simulated sequences results

To further ensure the robustness of the EKF thousands of different runs had to be performed.
Unfortunately, due to the excessive number of frames needed, an approach with real inputs coming from the RPEP was not feasible in the available time. To overcome this issue an alternative approach had to be pursued.
It consisted in the generation of pseudo-measurements given a GT and the error covariance matrix ($\boldsymbol{R}$) associated to the RPEP.
To this aim, exploiting the results obtained in Section 4.1.1, the $\boldsymbol{R}$ matrix could be obtained as:

$$\begin{gathered} \boldsymbol{R} = [r_{jk}] \\ \text{where} \\ r_{jk} = \frac{1}{N-1} \sum_{i}^{N} [(x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)] \end{gathered} \tag{4.1}$$

where x_{ij} represents the measurement of the quantity j at the iteration i and $\bar{x}_j$ is the GT of said measurement.

The overall distribution of the error covariance was then studied and a correlation with the distance can be seen in Figure 4.10:

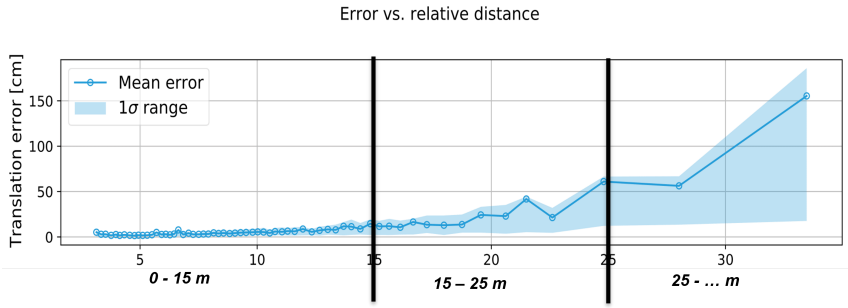


Figure 4.10: Mean and covariance of the error with respect to the relative distance

To account for this correlation, the relative distance was divided in three different sections, each with its own $\boldsymbol{R}$ matrix. The numerical values can be found in Appendix B

Once $\boldsymbol{R}$ was properly defined the testing procedure could start.
An initial random seed associated to a frame of SPEED was selected, thus, from the latter the initial conditions could be found. Then a high accuracy propagation provided the GT for the run, to which a random gaussian noise with covariance matrix $\boldsymbol{R}$ was added to simulate the measurements.

Once both the GT and the noise covariance matrix were obtained, the filter could start its iteration and the results could be extracted.

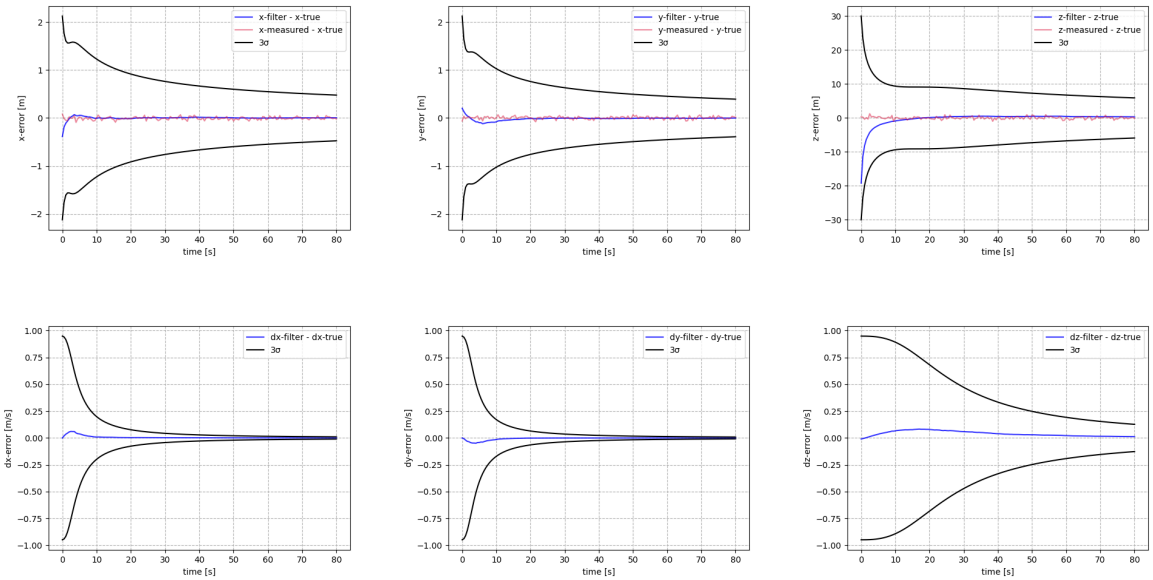


Figure 4.11: Generic results of an EKF run

Figure 4.11 reports the results of a generic run of the EKF. As can be clearly seen the filter quickly converges to a proper solution on both the relative separation components and their respective velocities.
It is also important to note that the z component and its associated velocity are always the last to reach convergence. This effect is caused by the higher uncertainty associated to that measurement.

Overall the filter proved to be extremely reliable and robust, never failing on over *1000* simulated runs. This performance is possibly due to the slower dynamics of the problem combined with the high accuracy of the measurements of the RPEP.

4.3 Unscented Kalman Filter

To ensure the reliability of the UKF as well, a procedure analogous to the one set for the EKF had to be performed.
Both tests on real and simulated inputs were performed showing good performance from the UKF, although a lower reliability was found.

4.3.1 Actual image sequence results

A set of tests to ensure the robustness of the UKF when subjected to real inputs had to be performed. To this aim the sequences analyzed in Section 4.1.2 were used.

Starting from the first sequence, the raw measurements obtained from the RPEP were fed to the filter and the output was thoroughly analyzed.
To facilitate the comparison between the raw measurement and the filtered state they have been both superimposed in the graphs. Moreover the 3σ line has also been visualized to easily discern any present patterns. The visualized results are visualized in Figure 4.12:

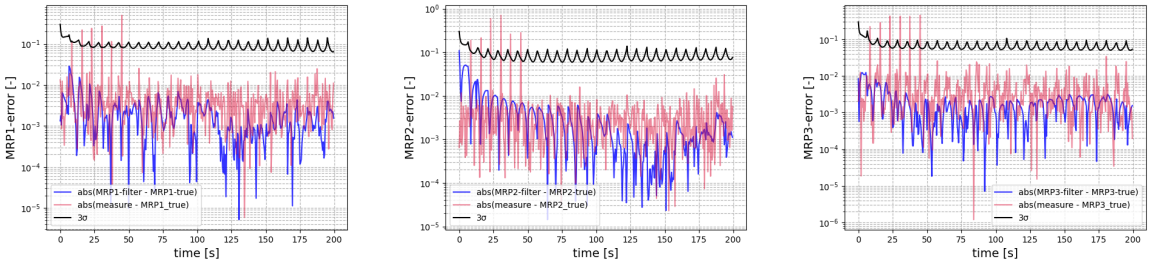


Figure 4.12: filtered MRP graphs of the *"Img000040"* sequence

Although no drastic improvement in the performance was present, the filter managed to successfully filter out any outlier, providing a clean and reliable output.
Overall the initial absolute angular error is of the 1 order, but then falls to the 10^{-2} *deg* order once the filtering action starts.
Focusing on the 3σ line, an oscillating pattern is easily visible throughout the graph. This behaviour was always present in the UKF output. This effect was more noticeable when the rotation rate was faster, presenting more frequent peaks, as such a dependence on the norm of the MRP was conjectured.
It can be considered that a faster rotation means that the MRP has to switch to its shadow set more frequently. As such, as this transformation is indeed non-linear, its application to the covariance implies a visible spike.
Furthermore, as the norm of the MRP grows, smaller initial uncertainties on the attitude translate to big increases of the MRP norm. Thus its uncertainty has to grow as it approaches unitary value and has to decrease after the switching to the shadow set as the norm tends to decrease.
This effect is clearly visible in the graph as the 3σ line presents a gradual increase, then a sudden spike when the switch to shadow is applied, and finally a gradual decrease.

Moving on to the second sequence, it presented a much slower rotation rate and thus a

slower evolution of the attitude. This was initially seen as a simplified case to perform an initial test of the filter, but, due to the numerous consecutive outliers measured by the RPEP, it proved to be a more challenging test case. The obtained results are visualized in Figure 4.13:

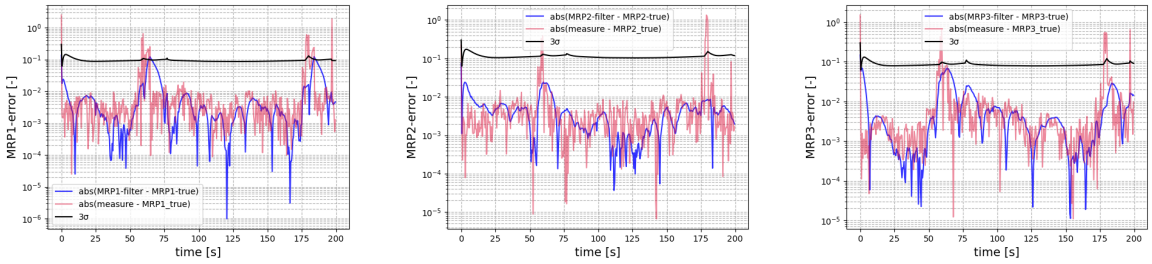


Figure 4.13: filtered MRP graphs of the *"Img000051"* sequence

As can be seen from the graphs in Figure 4.13, the filter presents two regions where a drastic increase of the error can be seen. This effect is indeed caused by the higher concentration of outliers in such regions.
More specifically, the UKF can identify and scrap such instances through the use of a *Mahalanobis distance* based process. This works very well when dealing with isolated cases, but, when dealing with dense regions with a high concentration of outliers, the filter can have problems.
This happens as, when a measurement is scrapped, the update phase of the UKF is skipped and the filter relies exclusively on the obtained propagation. When more measures are scrapped the filter virtually works "blind", as such an increase of the error is to be expected.
Nonetheless, the filter managed to dampen the intensity of the error in the worst regions, while quickly converging to smaller errors and better performances elsewhere.

In the end the UKF proved to be fit in dealing reliably with the real inputs provided. However, further extensive analysis was needed to ensure its performance and reliability and to identify the most common failure points.

4.3.2 Simulated sequences results

As was done for the EKF, a testing campaign on simulated inputs had to be performed on the UKF as well.
Following a similar process the $\boldsymbol{R}$ matrix had to be computed from the results of Section 4.1.1 using the Equation 4.1.

However, a different study of the error covariance had to be performed, as correlations with the norm of the MRP or the relative distance had to be investigated.

Starting from the latter, as can be seen from the scatter plots on Image 4.14, no significant relation between the error covariance and the relative distance is highlighted.

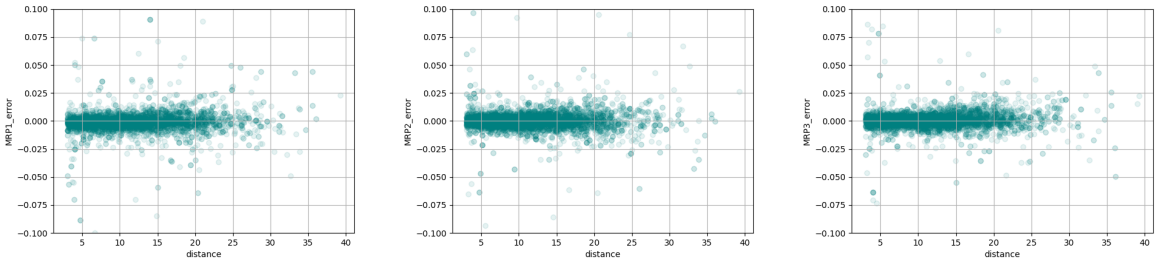


Figure 4.14: Scatter plots highlighting the MRP error with respect to relative distance

It has been speculated that this results was heavily influenced by the RPEP, as, before the LRN step the BB detected by the SLN is cropped and resized to a common resolution. Thus, virtually equalizing the field and removing all information on the distance (as closer frames lose resolution due to reshaping).
As the LRN receives such inputs, this "equalization" is then reflected on the attitude effectively removing its dependance on the distance.

Further analysis had to be performed to find eventual correlations with the MRP, as smaller errors on higher norms may lead to big discrepancies with the attitude as highlighted in Section 2.1.1.

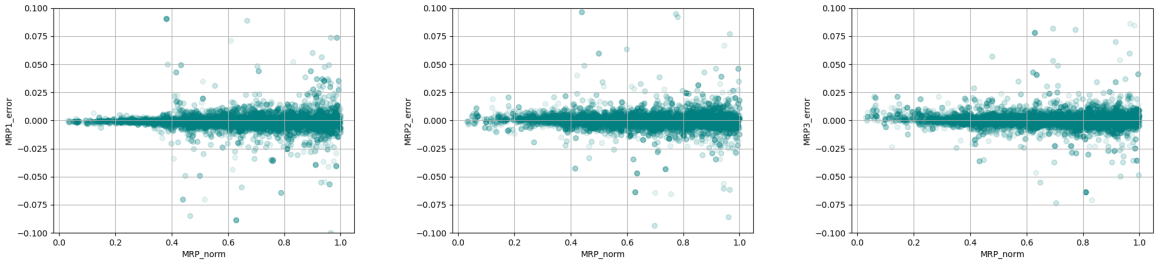


Figure 4.15: Scatter plots highlighting the MRP error with respect to its norm

As can be seen from the plots in Figure 4.15 a weak increase in the spread of the error is evident above a *0.4* norm. This can also be highlighted by considering the quaternion error spread in Figure 4.16

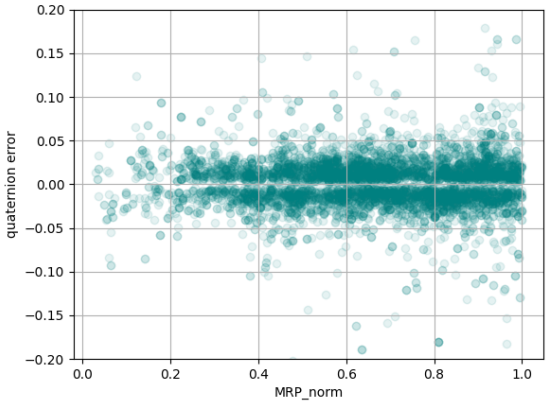


Figure 4.16: Scatter plot highlighting the quaternion error with respect to the MRP norm

To adopt a conservative approach to the problem, it was chosen to adopt a single noise covariance matrix ($\boldsymbol{R}$) evaluated only for MRP measurements above the *0.4* norm threshold. Its numerical value can be found in Appendix B

The computation of the $\boldsymbol{R}$ matrix enabled the start of the testing campaign on the UKF. Analogously to the EKF a random seed was generated, from which the initial relative attitude was extracted. Additionally an initial relative angular velocity had to be generated.
To this aim the analysis of a database containing various rotational periods (t_{rot}) obtained from the lightcurves of LEO tumbling objects was performed [Uni21].
A preliminary attempt at fitting a χ^2 distribution function from the obtained data was performed as shown in Figure 4.17. From this a range for the considered rotational periods was extracted and used throughout the testing phase.

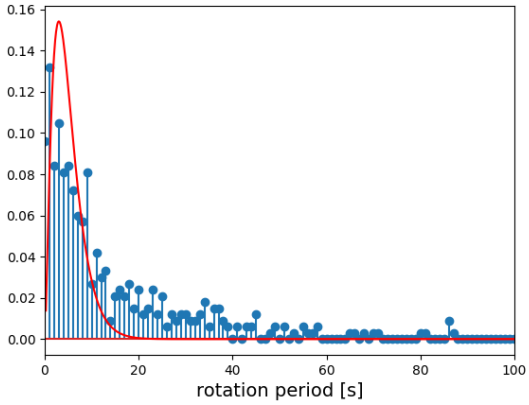


Figure 4.17: t_{rot} distribution with fitted χ^2 curve

Note that, as a rough first approximation, a χ^2 distribution has been chosen over the more common gaussian to avoid the possibility of a *0s* t_{rot} which would have been physically impossible.

Afterward, a relative angular velocity vector had to be defined, and, to do so a random unit vector was generated and then multiplied by $2\pi/t_{rot}$ where t_{rot} represents the chosen rotational period.
During the testing phase it was chosen to keep the rotational period above 5s as it was deemed a good compromise between performance and representation of the distribution.

Once the setup was completed the UKF could be executed. The obtained results are reported in Table 4.6:

UKF testing results			
t_{rot}	***performed runs***	***failed runs***	***fail rate***
15 s	*1000*	*24*	*2.4 %*
10 s	*1000*	*39*	*3.9 %*
5 s	*1000*	*83*	*8.3 %*

Table 4.6: Results from the UKF testing campaign

Note that the results have been partitioned with respect to their t_{rot}. This has been done to highlight the strong trend which correlates faster rotation with poorer performance. Moreover, all the presented runs have been propagated considering a timeframe of $500s$ and an acquisition rate of 2 fps.

Even though the UKF performance can be deemed acceptable, an analysis of the most common failure modes has been pursued to identify possible improvement fields.

4.3.3 Failure mode analysis

A detailed analysis of the most common failure modes and their triggers have been carried out.
An extensive number of runs have been performed in order to isolate those cases and analyze their internal states, in order to circle out the underlying triggers. To make the failures more common, a faster t_{rot} of $2s$ has been employed. This created a 53% failure rate confirming once again the correlation between faster rotation and failures.
Overall 200 runs have been performed and four main types of failure modes have been detected and classified.

To help the comparison between what is to be expected of a successful run and what was deemed a failed run, a reference case has been provided below:

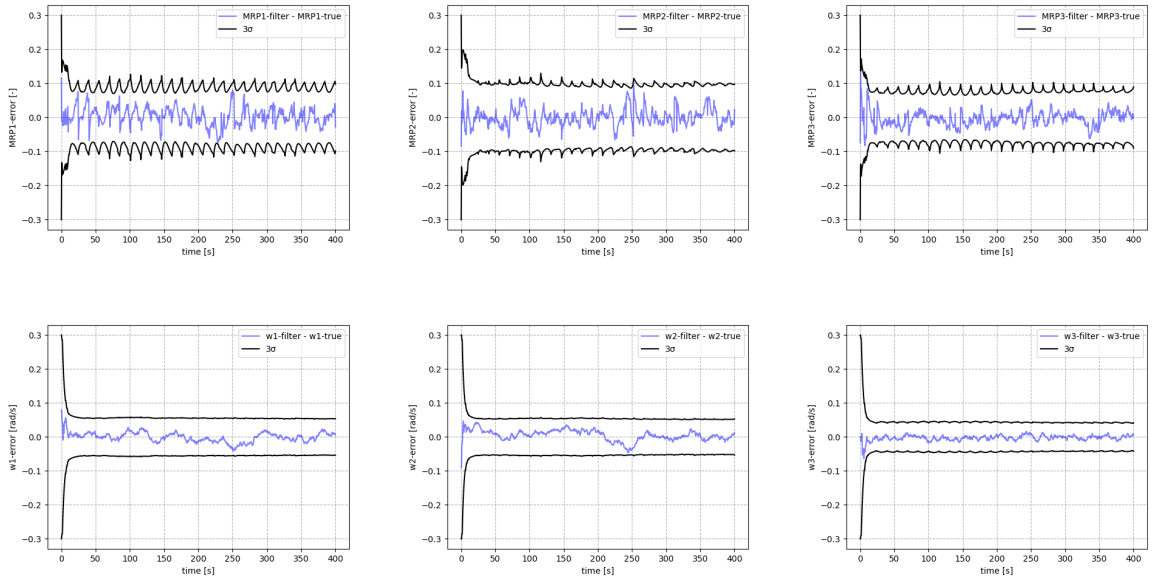


Figure 4.18: Generic results of an UKF run

The first and most common failure mode was defined as *Localized Spike*. As can be seen from the graphs of Image 4.19, it consists in an isolated sudden increase of the error followed by its instantaneous falloff.
The event happens in proximity to a shadow switch and is often caused by a detection of an outlier in those conditions.
Unfortunately no further information was found and no clear and correct explanation can be provided.
It is also to be noted that all the provided graphs are comparing the error related to both the real and the shadow set of the MRPs and are plotting the minimum of the two.

The *Localized Spike* is by far the most common failure mode and single-handedly adds up to around 65% of the detected fails. Moreover it is often found in combination with other failure modes, making the previous estimation lower than its actual appearance rate.

Although this failure mode can be considered an isolated case, and as such may not be a complete failure of the UKF, it was chosen to still identify and study it, as a solution may further enhance the reliability of the pipeline.

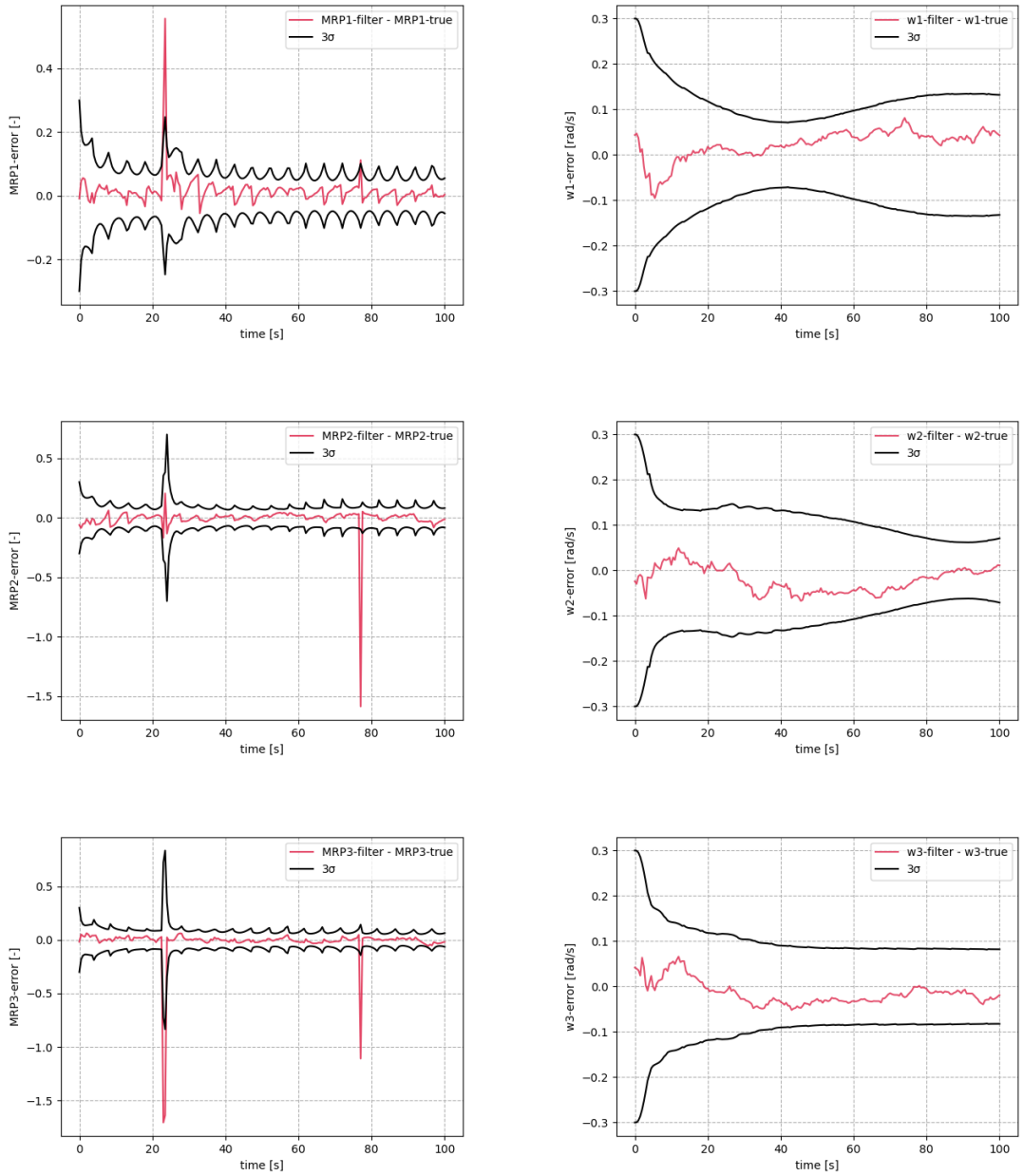


Figure 4.19: Identified *Localized Spike* (MRP [right], ω [left])

Moving on to the second identified failure mode, it has been named *Extended fail* as it lasts for a significant time frame. On the provided graphs on Image 4.20 it appears as a region where the error is oscillating at a higher amplitude.

From the study of the *Mahalanobis distance* evolution it was noted that a sequence of concentrated outliers triggered the initial error spike, thus the UKF discarded all those measurements but started to converge to a different solution.

Consequently, all successive measurements were treated as outliers as they had a very

high *Mahalanobis distance*. This happened as the filter expected measurements drifted far from the real ones.
After a while some measurements fell below the *Mahalanobis distance* threshold, thus allowing a slow convergence to the original solution.

Overall this failure mode accounts for around 30% of the detected fails.

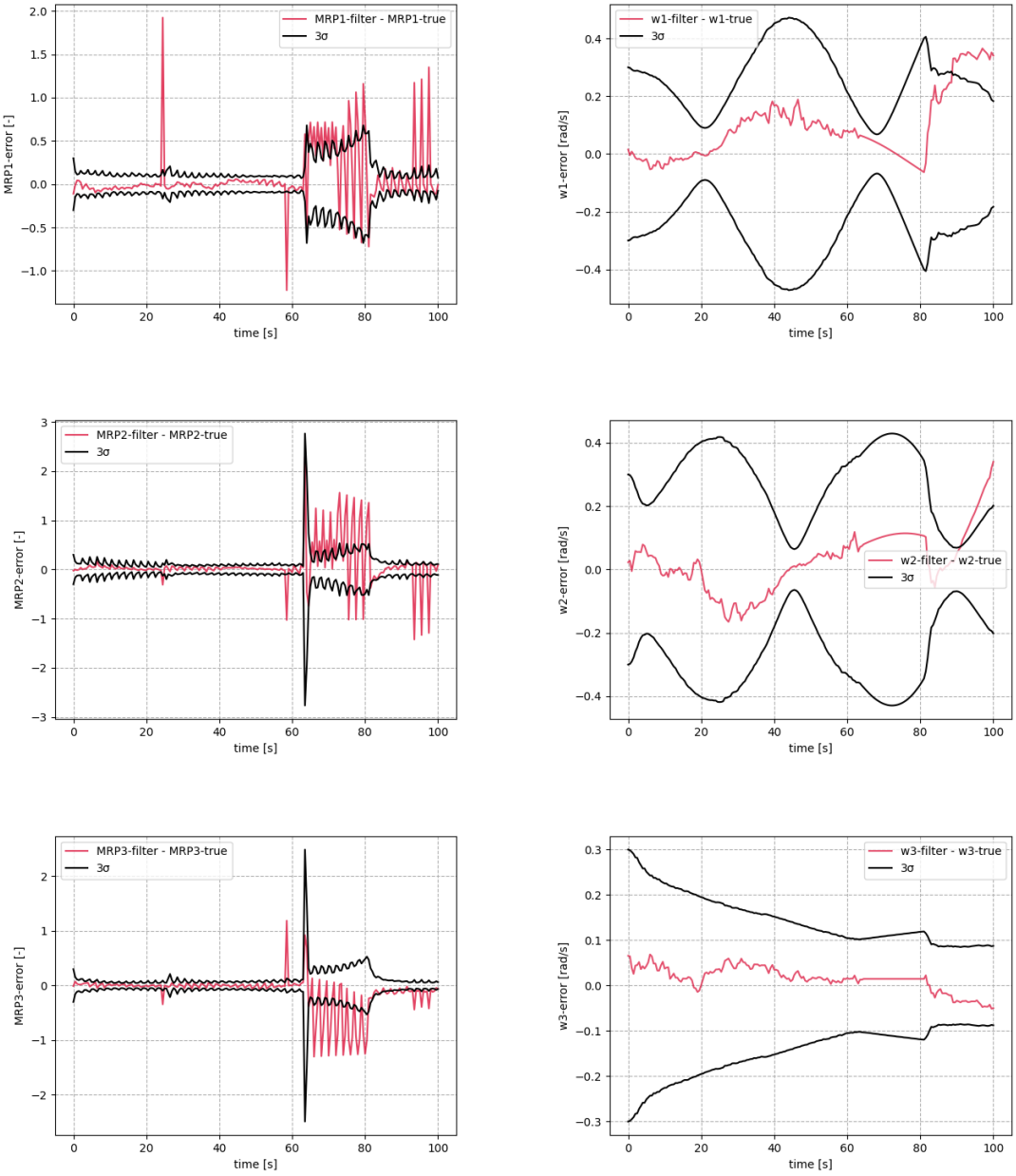


Figure 4.20: Identified *Extended fail* (MRP [right], ω [left])

A particular type of *Extended fail* has been identified and has been named *Initial Fail*. As can be seen from the graphs of Figure 4.21, this failure happens at the start of a sequence and carries on for a while.
By looking at the data it was evident that a bad initial guess caused by an outlier made the UKF converge on a wrong solution right away.
Similarly to the Extended fail, the following measurements were treated as outliers and discarded up until some measure managed to enter the filter and return it to convergence.

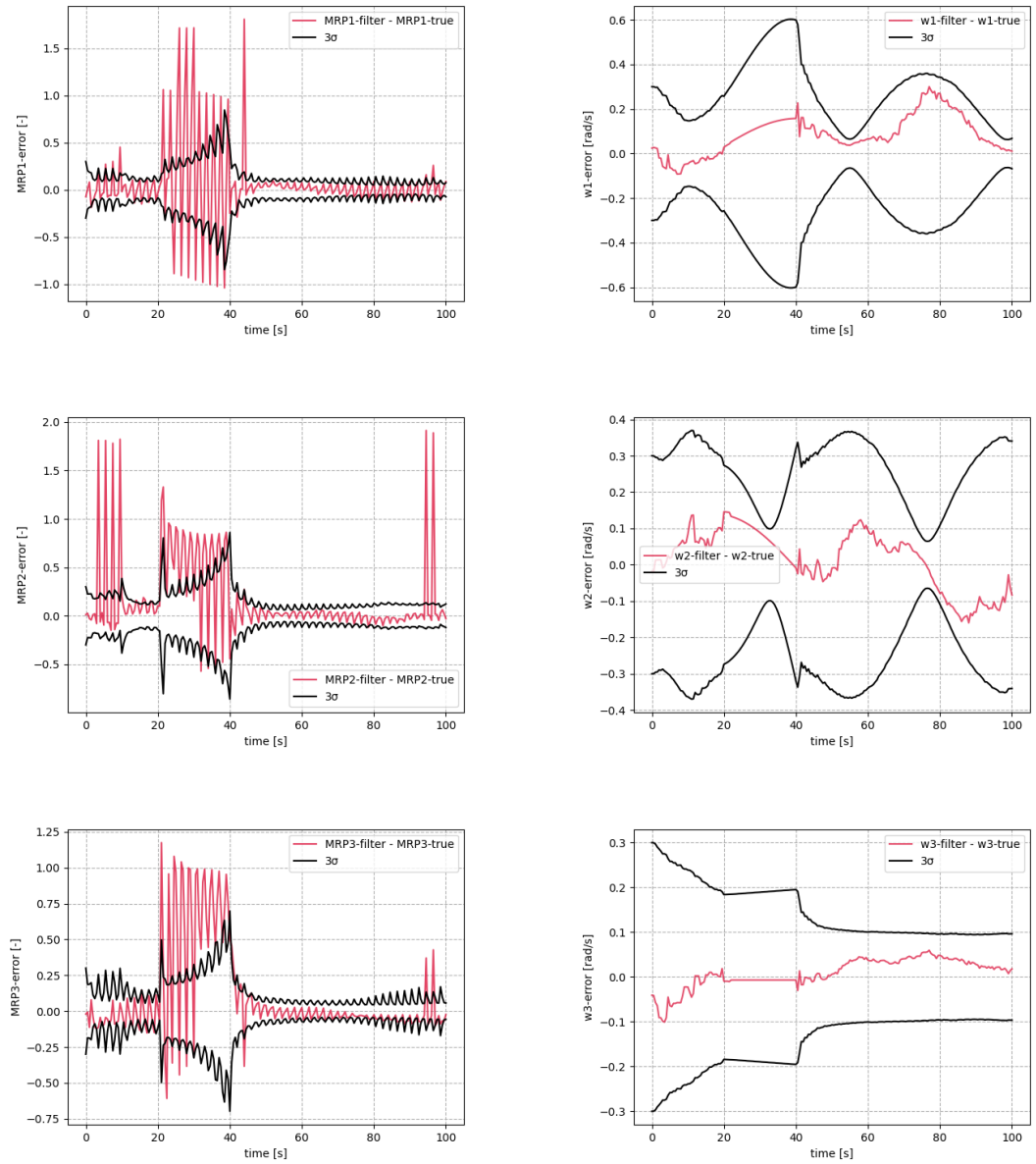


Figure 4.21: Identified *Initial fail* (MRP [right], ω [left])

The final failure mode and also the less common is defined as *Total fail*. As can be seen from the graphs on Image 4.22 it implies the complete failure of the run.
The underlying causes of this failure are the same as those of the *Initial fail*, with the main difference being that the filter does not manage to converge in a reasonable time.

Fortunately this failure mode is the rarest accounting only for around 5% of the total detected failures.

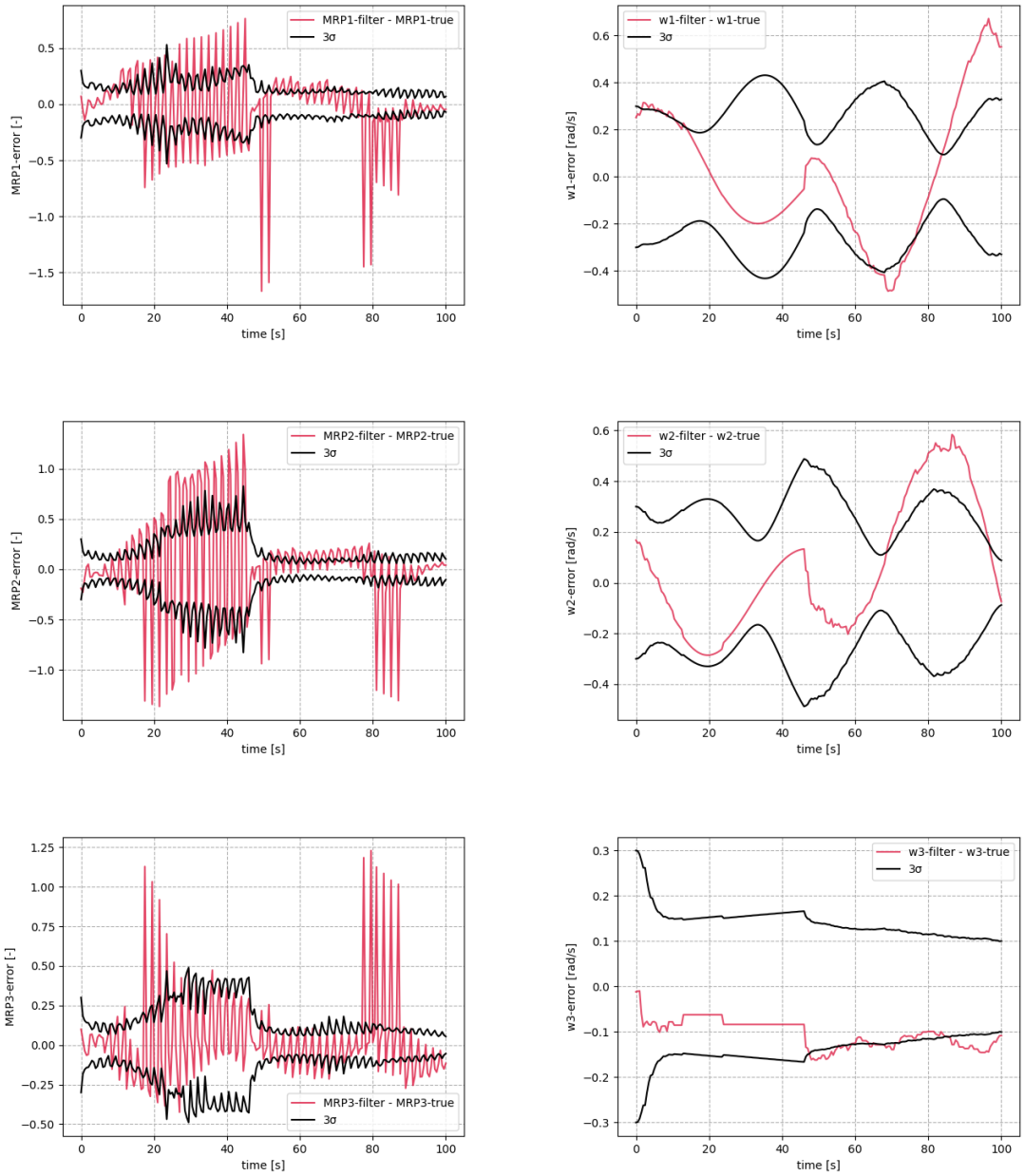


Figure 4.22: Identified *Total fail* (MRP [right], ω [left])

Some preliminary means of failure avoidance have been implemented in the UKF in the form of a *Mahalanobis distance* based correction.
To avoid the detected failures more sophisticated countermeasures have to be employed. Those could range from more elaborated *Mahalanobis distance* based correction procedure, to planned total filter resets.

4.4 Raspberry Pi implementation

A preliminary implementation of the pipeline on a platform which is more representative of processors available onboard a spacecraft, has been carried out and the results have been noted.
The platform chosen for this task was the *Raspberry Pi 4 2GB* single board computer running the *Raspberry Pi OS 64bit* operating system, a modified version of *Linux*.
Strong drivers for the use of this platform were mainly: relatively good performance, built-in *Python* support and a vast amount of literature.
In addition, similar platforms have already been deployed on real spacecrafts such as *NASA's Seeker-1* cubesat, which mounted a *SBC Intel Joule* single board computer [BA19].

Both the RPEP and the filtering pipeline have been implemented and executed with promising results.

4.4.1 Hardware performance

Starting from the RPEP, its implementation was quite tricky as the "wheel" files for the core *Python* packages required were not easily accessible on the Raspberry platform.
Furthermore, once all packages were successfully installed, the pipeline was limited to only a couple dozen images for each run. This limitation was put in place to avoid RAM saturation issues, overheating of the hardware and excessive computational time.
Overall the test was performed on 100 different frames taken from the simulated database, the resulting mean computational load is therefore reported in Table 4.7 for each step:

RPEP performance			
SLN	***LRN***	***EPnP***	***RPEP total***
5.21 s	*87.22 s*	*0.28 s*	*92.71 s*

Table 4.7: RPEP performance on the *Raspberry Pi 4 2GB*

Moving on to the filtering procedure, its implementation was more straightforward without significant issues. Multiple fully simulated runs were performed with propagation times of *1000s* at *2 fps*.
The timing reported was considered between the start of the prediction step and the end of the update step.
The observed computational loads are presented in Table 4.8:

Filters performance		
EKF	***UKF***	***Filters total***
0.02 s	*0.37 s*	*0.39 s*

Table 4.8: filtering pipeline performance on the *Raspberry Pi 4 2GB*

As can be clearly seen the RPEP predictably presents the higher computational load being around two orders of magnitude greater than the filter load.
This shortcoming was mainly due to the un-optimized nature of the algorithm which ran on Python with little focus on reducing its computational footprint.
It is reasonable to expect an improvement of up to two orders of magnitude once proper optimization procedures are put in place. These can range from compiling the algorithm on the C platform (which can improve the performance by a factor of 10) or performing proper FPGA hardware optimization.
Furthermore the exploitation of dedicated analog matrix processors [Elb+21] or the adoption of pruning and quantization techniques [Lia+21] could further enhance the performance of the NN section.
Finally, even though the *"Raspberry Pi"* platform may be indicative of the performances to be expected from a single board computer, space hardware that needs to run CNN driven operations is bound to present improved performances to cope with the applied load, and thus, in a real scenario the computational time will be further decreased.

Overall, even though the obtained results can seem very poor, a lot of room for improvement is left and therefore, the possibility of spaceborne applications of the proposed pipeline is not to be discarded yet.

5 | Conclusion and future work

5.1 Conclusion

This thesis tackled the problem of the relative attitude determination of an uncooperative spacecraft starting from a pre-existing CNN based Relative Pose Estimation Pipeline (RPEP).
To fully complete the process a way to produce a vast amount of different spaceborne images and a way to filter the RPEP output was needed. Thus the contribution of this dissertation is to provide and test both procedures.

Starting from the spaceborne imagery generation procedure, a *Blender* based pipeline was designed and implemented.
It is capable of generating collections of single uncorrelated frames from a given baseline or rendering coherent sequences of images given proper initial conditions and fps rate.
The pipeline can also simulate various lighting conditions based on the orbital motion of the studied spacecrafts and the considered season.

To ensure its reliability and to validate the photorealism of the produced frames, a testing campaign was performed.
To this aim the SPEED database was extensively used as a reference. At first simple visual inspection was performed to allow the tuning of both texturing and environmental conditions. Then, once the resulting frames were deemed satisfactory, the CNN based RPEP devised by Massimo Piazza was exploited to further validate the proposed image generation procedure.
Overall a good degree of similarity was reached, proven also through a comparison with the whole SPEED database.
Once the image generation procedure was successfully validated, some sequences were generated, then evaluated by the RPEP and finally a filtering procedure was employed to further enhance the results.

Focusing now on the filtering procedure, Kalman Filter (KF) were extensively used throughout the dissertation due to their perfect fit for the problem at hand.

To address the *relative separation problem* it was chosen to employ an Extended Kalman Filter (EKF) thanks to its simplicity coupled with the linearity of the considered dynamics.
Instead, for the more complex *relative attitude problem* an Unscented Kalman Filter (UKF) was used due to the non-linearity of the dynamics and their faster evolution.
As previously mentioned the whole filtering pipeline was tested using real frames from the image generation procedure. However, as a thorough study of the robustness of the employed filter was pursued, a way to artificially and quickly simulate the desired inputs was needed.
To this aim the characterization of the error covariance of the RPEP was performed in order to enable the artificial generation of proper inputs.

Overall the filtering pipeline was extremely successful with the EKF performing flawlessly on all the tests performed, and the UKF only failing 8.3% of the times on the worst case analyzed.

Finally, the implementation of the pipeline on a *Raspberry Pi 4 2GB* single board computer was carried out with the aim of testing it on more significant hardware. Overall the obtained results were very poor but, due to the unoptimized nature of the pipeline, a lot of room for improvement is available and leaves hope for eventual future embedded implementation.

5.2 Future work

Some suggestions on the steps to be performed to further improve the work outlined in this thesis are here proposed:

- The optimization of the scripts is of paramount importance to run the pipeline on space-grade components, as such low level C/C++ implementation is needed.
- Looking at the detected UKF failure modes, proper countermeasures have to be devised to ensure stronger robustness and reliability.
- Once the pipeline has been properly optimized, a "hardware-in-the-loop" testing campaign employing space-grade hardware and real cameras should be pursued.
- A way to generalize the pipeline and allow the study of spacecrafts other than *TANGO* should be pursued.

Bibliography

[Kal60] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45.

[LMS82] E.J. Lefferts, F.L. Markley, and M.D. Shuster. "Kalman Filtering for Spacecraft Attitude Estimation". In: (1982).

[Dho+89] M. Dhome et al. "Determination of the attitude of 3D objects from a single perspective view". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11.12 (1989), pp. 1265–1278.

[JUD95] S.J. Julier, J.K. Uhlmann, and H.F. Durrant-Whyte. "A new approach for filtering nonlinear systems". In: 3 (1995).

[SJ95] Hanspeter Schaub and John Junkins. "Stereographic Orientation Parameters for Attitude Dynamics: A Generalization of the Rodrigues Parameters". In: *Journal of the Astronautical Sciences* 44 (Jan. 1995).

[MG00] Oliver Montenbruck and Eberhard Gill. "Satellite Orbits: Models, Methods and Applications". In: (2000).

[WV00] E.A. Wan and R. Van Der Merwe. "The unscented Kalman filter for nonlinear estimation". In: (2000).

[MG02] O. Montenbruck and E. Gill. "Satellite Orbits: Models, Methods, and Applications". In: *Applied Mechanics Reviews* 55.2 (Apr. 2002).

[SJ03] Hanspeter Schaub and John L Junkins. *Analytical mechanics of space systems*. Aiaa, 2003.

[KT08] Hans Kruger and Stephan Theil. "TRON - Hardware-in-the-Loop Test Facility for Lunar Descent and Landing Optical Navigation". In: (2008).

[KS09] Christopher D. Karlgaard and Hanspeter Schaub. "Nonsingular Attitude Filtering Using Modified Rodrigues Parameters". In: *Journal of the Astronautical Sciences* (2009).

[Air11] Airbus. *SurRender Software*. 2011. URL: https://www.airbus.com/en/products-services/space/exploration/moon.

[Gav13] Henri P. Gavin. "The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems". In: 2013.

[DBJ14] Simone D'Amico, Mathias Benn, and John Jørgensen. "Pose estimation of an uncooperative spacecraft from actual space imagery". In: *Int. J. of Space Science and Engineering* (Jan. 2014), pp. 171–189.

[MZ14] Mauro Massari and Mattia Zamarro. "Application of SDRE technique to orbital and attitude control of spacecraft formation flying". In: *Acta Astronautica* 94.1 (2014), pp. 409–420.

[Red+16] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[Ree+16] Benjamin Reed et al. "The Restore-L Servicing Mission". In: *AIAA space 2016*. (2016), p. 5478.

[BGF17] Facundo Bre, Juan Gimenez, and Víctor Fachinotti. "Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks". In: *Energy and Buildings* 158 (Nov. 2017).

[Opr+17] Roberto Opromolla et al. "A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations". In: *Progress in Aerospace Sciences* (2017).

[SVD18] Sumant Sharma, Jacopo Ventura, and Simone D'Amico. "Robust Model-Based Monocular Pose Initialization for Noncooperative Spacecraft Rendezvous". In: *Journal of Spacecraft and Rockets* 55 (6 Nov. 2018), pp. 1414–1429.

[ZXB18] Gaopeng Zhao, Sixiong Xu, and Yuming Bo. "LiDAR-Based Non-Cooperative Tumbling Spacecraft Pose Tracking by Fusing Depth Maps and Point Clouds". In: *Sensors* 18 (2018).

[AS19] ACT and SLAB. *Pose Estimation Challenge 2019*. 2019. URL: https://kelvins.esa.int/satellite-pose-estimation-challenge/home/.

[BA19] Brian Banker and Scott Askew. "Seeker 1.0: Prototype Robotic Free Flying Inspector Mission Overview". In: (2019).

[Che+19] Bo Chen et al. "Satellite Pose Estimation with Deep Landmark Regression and Nonlinear Pose Refinement". In: (Oct. 2019), pp. 2816–2824.

[ESA19a] ESA. *ESA commissions world's first space debris removal*. 2019. URL: https://www.esa.int/Safety_Security/Clean_Space/ESA_commissions_world_s_first_space_debris_removal.

[ESA19b] ESA. *Planet and Asteroid Natural Scene Generation Utility software*. 2019. URL: https://pangu.software/.

[SLA19] SLAB. *Kelvins, pose estimation challenge post mortem leaderboard*. 2019. URL: https://kelvins.esa.int/satellite-pose-estimation-challenge/leaderboard/post-mortem-leaderboard.

[Sun+19] Ke Sun et al. "Deep High-Resolution Representation Learning for Human Pose Estimation". In: (June 2019).

[Kis+20] Mate Kisantal et al. "Satellite Pose Estimation Challenge: Dataset, Competition Design and Results". In: (2020).

[Pia20] Massimo Piazza. "Deep Learning-Based Monocular Relative Pose Estimation of Uncooperative Spacecraft". MA thesis. Politecnico di Milano, Nov. 2020.

[PG20] Pedro F. Proença and Yang Gao. "Deep Learning for Spacecraft Pose Estimation from Photorealistic Rendering". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2020).

[ult20] ultralytics. *YOLO v5*. 2020. URL: https://github.com/ultralytics/yolov5.

[AS21] ACT and SLAB. *Pose Estimation Challenge 2021*. 2021. URL: https://kelvins.esa.int/pose-estimation-2021/challenge/.

[Bla+21] Kevin Black et al. "Real-Time, Flight-Ready, Non-Cooperative Spacecraft Pose Estimation Using Monocular Imagery". In: (2021).

[Elb+21] Mohammed Elbtity et al. "An In-Memory Analog Computing Co-Processor for Energy-Efficient CNN Inference on Mobile Devices". In: (2021), pp. 188–193.

[ESA21] ESA. *Imaging model satellite in space-like lighting conditions*. 2021. URL: https://www.esa.int/ESA_Multimedia/Images/2021/10/Imaging_model_satellite_in_space-like_lighting_conditions.

[Gru21] Northrop Grumman. *Space Logistics*. 2021. URL: https://www.northropgrumman.com/space/space-logistics-services/.

[Leb+21] Jérémy Lebreton et al. "Image Simulation for Space applications with the SurRender software". In: (June 2021). Presented at 11th International ESA Conference on Guidance, Navigation and Control System.

[Lia+21] Tailin Liang et al. "Pruning and Quantization for Deep Neural Network Acceleration: A Survey". In: (2021).

[ML21] Michele Maestrini and Pierluigi Di Lizia. "COMBINA: Relative Navigation for Unknown Uncooperative Resident Space Object". In: *AIAA SCITECH 2022 Forum*. 2021.

[Sco+22] Andrea Scorsoglio et al. "Image-Based Deep Reinforcement Meta-Learning for Autonomous Lunar Landing". In: *Journal of Spacecraft and Rockets* (2022).

[Tor22] Leo Torres. *Mars in UNREAL ENGINE 5: A UE5 Space Cinematic*. 2022. URL: https://www.artstation.com/artwork/X1nYxL.

[Swe] OHB Sweden. *OHB-sweden website*. URL: https://www.ohb-sweden.se/.

[Uni21] Kazan Federal University. *MMT lightcurve observation database*. 2006-2021. URL: http://mmt.favor2.info/satellites.

Appendices

A | Linearization of the relative separation problem

Note, to simplify the notation the parameter r_c has been adopted as:

$$r_c = [(\bar{r} + x)^2 + y^2 + z^2]^{3/2}$$

The components of the Jacobian matrix $\mathbf{F}$ are here reported:

$$\frac{\partial \ddot{x}}{\partial x} = \frac{\mu}{r_c^3}\left[3\left(\frac{\bar{r}+x}{r_c}\right)^2 - 1\right] + \dot{\nu}^2$$

$$\frac{\partial \ddot{x}}{\partial y} = 3\mu\left(\frac{\bar{r}+x}{r_c^5}\right)y - 2\dot{\nu}\frac{\dot{\bar{r}}}{\bar{r}}$$

$$\frac{\partial \ddot{x}}{\partial z} = 3\mu\left(\frac{\bar{r}+x}{r_c^5}\right)z$$

$$\frac{\partial \ddot{x}}{\partial \dot{y}} = 2\dot{\nu}$$

$$\frac{\partial \ddot{y}}{\partial x} = 3\mu\left(\frac{\bar{r}+x}{r_c^5}\right)y + 2\dot{\nu}\frac{\dot{\bar{r}}}{\bar{r}}$$

$$\frac{\partial \ddot{y}}{\partial y} = \frac{\mu}{r_c^3}\left[3\left(\frac{y}{r_c}\right)^2 - 1\right] + \dot{\nu}^2$$

$$\frac{\partial \ddot{y}}{\partial z} = 3\mu\left(\frac{y}{r_c^5}\right)z$$

$$\frac{\partial \ddot{y}}{\partial \dot{x}} = -2\dot{\nu}$$

$$\frac{\partial \ddot{z}}{\partial x} = 3\mu\left(\frac{\bar{r}+x}{r_c^5}\right)z$$

$$\frac{\partial \ddot{z}}{\partial y} = 3\mu\left(\frac{y}{r_c^5}\right)z$$

$$\frac{\partial \ddot{z}}{\partial z} = \frac{\mu}{r_c^3}\left[3\left(\frac{z}{r_c}\right)^2 - 1\right]$$

B | Error covariance matrices

The numerical values of the $\boldsymbol{R}$ matrix applied to the EKF are here reported with the corresponding distance range:

- ***0 - 15 m*** $= \begin{bmatrix} 5.04 \cdot 10^{-5} & 8,87 \cdot 10^{-6} & 2.17 \cdot 10^{-5} \\ 8,87 \cdot 10^{-6} & 1.05 \cdot 10^{-4} & 1.68 \cdot 10^{-4} \\ 2.17 \cdot 10^{-5} & 1.68 \cdot 10^{-4} & 9,60 \cdot 10^{-3} \end{bmatrix}$

- ***15 - 25 m*** $= \begin{bmatrix} 9.32 \cdot 10^{-4} & 1.68 \cdot 10^{-4} & -1.48 \cdot 10^{-3} \\ 1.68 \cdot 10^{-4} & 7.92 \cdot 10^{-4} & 2.77 \cdot 10^{-3} \\ -1.48 \cdot 10^{-3} & 2.77 \cdot 10^{-3} & 1,60 \cdot 10^{-1} \end{bmatrix}$

- ***25 - ... m*** $= \begin{bmatrix} 3.04 \cdot 10^{-3} & -1.35 \cdot 10^{-3} & 9.08 \cdot 10^{-3} \\ -1.35 \cdot 10^{-3} & 6.55 \cdot 10^{-3} & 5.90 \cdot 10^{-2} \\ 9.08 \cdot 10^{-3} & 5.90 \cdot 10^{-2} & 1.63 \end{bmatrix}$

The numerical values of the $\boldsymbol{R}$ matrix applied to the UKF are here reported:

$$\boldsymbol{R} = \begin{bmatrix} 0.00833 & -0.00023 & -0.001 \\ -0.00023 & 0.0138 & 0.00067 \\ -0.001 & 0.00067 & 0.00637 \end{bmatrix}$$