Università
Ca'Foscari
Venezia

Master's Degree Course: Computer Science
Class LM-18

Application of different types of Kalman Filter to approximate state of an UAV Model

A thesis affine to the course of
3D Computer Vision

Grad Student:
Alberto Carraro
Mat. 855255

Supervisor:
Filippo Bergamasco

Accademic year 2020/2021

Index

Abstract

The invention of Kalman filter brings to the engineer world an important solution on the noise management, and more in general, the possibility of smoothing perturbed data collected from a sensor.
So far, studies and upgrades adopted in this environment helped researchers to develop different kind of filters, adopting different methods and mathematical tools which allowed to handle different kinds of data and perform better.
The aim of this thesis is study the most used Kalman filters, and in particular the combination of them with algebraic structures such as Dual Quaternions, in order to establish, given a set of data coming from an UAV model, which of them performs better, giving the best approximation on results gained from measurements and getting the best ideal trajectory.
Data are withdrawn from an FPV drone, by recording a flight session and saving logs in the built-in black-box. These are then parsed, transformed in a csv file and only relevant data, such as axis rotations, GPS coordinates and altitude, are used to perform our experiments.
Several plots were generated to help us to made a comparison about results obtained from our study, concluding that the Unscented Kalman filter with Dual Quaternions is the best choice to dealing with these kind of data.

1 Introduction

1.1 Overview

Unmanned aerial vehicles, also called UAVs, have attracted in the last years lot of interest because they can be used in many different scenarios and situations, and bring the possibility to plan mobile and smart missions. Actually they are employed in researchers regarding Artificial intelligence, robotics [43, 27, 21], to make it possible autonomous vigilance, inspections and monitoring strategies [30, 8].
Tracking and filtering are well-studied problems in this context, and during the years several approaches have been developed. Many methods were used to predict the state of UAVs, such as Least Square [13, 41], Gauss-Newton [14], and Kalman filters [28, 48, 16]. This last method is widely used, and many works on that have been developed.
This thesis is intended to provide a comparison between Linear Kalman filter and Unscented Kalman filter, both adopted to estimate the position and the trajectory of an FPV (First Person View) drone. The work is based on testing and studying the filtering method and in particular, Linear and Unscented Kalman filters will be compared in some experiments dealing with raw data coming from a flight session of the drone.
Let's begin from the history and a little background on the theory on this topic.

1.2 Related Works

1.2.1 Kalman as filtering method

Many researches have been conducted on the study of the prediction by statistical instruments to determine the state of an object moving around during a time t. Kalman Filter is one of this methods and it is one of the most used and finds its usage in different applications. When we talk about Kalman filters we are dealing

with an algorithm which makes the history on Engineering's world.

From scratch, this algorithm is based on the estimation of a non observable hidden state, which belongs to a set of random variables, by basing its prediction on the previous state already computed. These variables, that in real world are measurements taken from sensors, are generally subjected to an additive Gaussian noise, taken into account during measurement by a covariance matrix.
The filter was developed for the first time from Emigré Rudolf E. Kálmán, with the contribution of Thorvald Nicolai Thiele and Peter Swerling, but in general, the attribution of the merit on the the development of the algorithm was given to Stanley F. Schmidt. In fact, he discovered that the algorithm could be divided into two distinct parts, in which, the first one adopted for computing time periods between sensor outputs and the other for updating measurements.
The first true applications of the filtering method was in the Apollo program, by using it on the main navigation computer for the computation of landing trajectories. After that event, Kalman filters have been adopted in the navigation systems of US Navy nuclear ballistic missile submarines and in the guidance and navigation systems of cruise missiles such as the US Navy's Tomahawk missile and the US Air Force's Air Launched Cruise Missile. They were also employed in reusable launch vehicle guidance and navigation systems, as well as the attitude control and navigation systems of spacecraft docking at the International Space Station.

Another important result was given by later studies conducted by the Dynamic Analysis Branch of NASA having as director Dr. Schmidt, to the discovery of the today's EKF (Extended Kalman Filter). Thanks to these researches and their first major application, the following was achieved [32]:

1. it has been proved the adaptability of the original Kalman theory to nonlinear problems
2. The development of an extension of Kalman Filter (EKF), which reduces the effects of the problems in nonlinear systems after a linearization over the best real state estimation
3. A new formulation of the original Kalman algorithm in separate time-update and measurement-update parts so measurements could be processed at any arbitrary time interval
4. Give an example of the possibilities of the Kalman filter through its application in a full digital simulation on a spaceship in which have been solved nonlinear orientation and navigation problem

5. The disclosure of results about the simulation for the possible incorporation into the control and navigation system of the Apollo spacecraft

6. The dissemination of data of the Ames Kalman filter to lot of scientific and aerospace's units through presentations and formal work

Both the Kalman Filter and the Extended Kalman filter were employed in robotics, where they are used to deal with trajectory tracking, manipulator robot position estimation, SLAM (Simultaneous Localization and Mapping), and object identification, among other things, depending on the model's linearity or non-linearity.
Many researches have found that the EKF presents many problems when the uncertainty becomes an important factor: An example is given on [25] in which the EKF was used in the context of Visual Odometry, and it was proved that in the application of the algorithm to a SLAM system proving that EKF is inconsistent due to Jacobians computation. Moreover it's shown that the observability properties of the EKF's linearized system do not match with those of the underlying system. The same reasoning is also underlined in [4]. Most of the works on Extended Kalman filter haven't use it without any correction or extension, in fact in [39], as in many other authors, created a variant, which in this case was based on Gauss–Newton iterative solution of a nonlinear smoothing problem at each sample time.

After that, many versions of Kalman filter have been implemented, such as the UKF (Unscented Kalman filter) which is based on the Unscented Transformation [15]. The benefits which it brings are not negligible, and actually is one of the most used filter: it adopts the principle which a set of sample points are used to correct mean and covariance by ensuring same performance of the linear counterpart.

As we have seen the Kalman Filter is an optimal estimator, and some studies have shown that it has a lot of applications, concerning also chemistry researches [6]. In particular, since the aim of this thesis is the application of the Kalman Filter for UAV trajectory estimation, we will explain, later on, that its application with mathematical instruments such as Dual Quaternions, will bring optimal results.

1.2.2 Modelling with Manifolds

Many papers threat this topic related with the use of image processing and filtering techniques : in particular [35] in his studies on tensors (a positive definite symmetric matrices), revealed that Riemannian metric is a very powerful tool to generalize the statistics on manifolds, and demonstrate that tensor field can be generalized for many different geometric algorithm such filtering, diffusion, interpolation.
Regarding filtering method, in [12], they provide an experimental filter in which uses both Unscented Kalman filter and Riemannian geometry, concluding that sigma

points perfectly fits on the Riemannian extension and that can be applied in many problems.
In this work, as already said, we are going to use Dual Quaternions and, since we are working with Unit dual Quaternions, we know for sure that they belong to a Riemannian Manifold.
Some references and backgrounds of this topic are taken from [34, 12, 9].

1.2.3 Dual Quaternions

Unit Dual Quaternions, but in general Quaternions, as we will see, are mathematical instruments in which we find complex numbers and they are useful to provide both rotation and translation in a unique notion [11], and find a valid alternative to rigid transformations computed by matrices or pairs.
They are applied on Geometric 3D computer vision, games graphic as cited in [19] or in [42] the Unscented Kalman filter is used in Combination with unit Dual Quaternions to study pose estimation of a 3D model, in bio-mechanics environments as demonstrated in [37] and, in general, in most common cases on theoretical kinematics. In the beginning dual Quaternions where called bi-Quaternion on the first article in which they have been described for the first time [1].
As mentioned in [38], position and orientation can be represented in many different ways, such as matrices of homogeneous transformations, Euler angles, unit Quaternions or dual Quaternions. In particular when we are dealing with Euler angles, since they have a rotation in sequence nature, the Euler method is not quite effective with multiple orientation path on trajectory tracking. Funda and Paul in one of their own script [10] showed that dual Quaternions are the most efficient and compact approach to describe a screw displacement when comparing screw theory expressions in terms of representation, storage, and computing efficiency.
The unit Quaternion representation, according to Funda, Taylor, and Paul, offers the same advantages but the dual Quaternion representation was chosen because it provides a better approach to describe both rotational and translational transformations in a single representation.

Those concepts are all referring to the theory of the screw motion and Chasles's theorem.
Rigid body motions, which include both rotation and translation, can be characterized by elements belonging to the special Euclidean group SE(3). As mentioned in [23], for the estimation of SE(3) there are many ways to estimate hidden states that belongs to it.
For example it's known that spatial rotations suffers about discontinuities or singularities in certain situations. Some optimizations and resolutions about this problem have been provided, but the main resolutions were founded by using dual Quaternions.

The above described work, is the most similar of the one described in this thesis: the use of Dual Quaternions to describe the rigid motions and the manifold geometry, are similarly adopted here with the Unscented Kalman filter, to predict the hidden state of an UAV.
Many other related works have been focused on this topic, because it's a basic theory on the development of Kinematics systems and computer graphics:
An article studying inertial navigation system [49], have proposed a novel algorithm based on the use of Dual Quaternions to deal with the principle of inertial navigation. In fact it has been shown that the this principle can be represented by three kinematic equations in dual Quaternion form, solving them by creating a novel algorithm by obtaining more accuracy than any other conventional algorithm for high precision navigation systems.

Many other related works, adopt Dual Quaternions for mapping points on $SE(2)$ and $SE(3)$ which are both Special Euclidean Groups. The first group is used to dealing with planar rigid body motion, instead the other for Rigid motions on 3D euclidean space.
Instead in [24] it has been presented a new SLAM approach with the use of Dual Quaternions for a better understanding of the non linear structure of $SE(2)$, and handle better the system uncertainty. Moreover it confirms that UKF and Dual Quaternions are the best tools to deal with some correlated problems.
Another example is given from [44], in which dual Quaternions are used for a multi-view registration algorithm which has the purpose to project alignments into a common reference frame. Here Dual Quaternions represent the motion and the 3d transformations and the diffusion along the graph, which in part is similar on what we're going to do on our work.
On a similar topic [22], in which to stochastically describe uncertain 6-DoF rigid body movements represented by unit dual Quaternions, an unique technique is provided. The resultant distribution is defined directly on the manifold of unit dual Quaternions, with no local linearizations, and takes into account the underlying group's non linearity. Furthermore, by applying the Riemannian geometry parallel transport approach to the Bingham distribution on the $\mathbb{S}^3$ hypersphere, a probabilistic interpretation of the connection between the rotation and translation components is allowed.

In this work, the use of Dual Quaternions is applied to find out trajectory estimation of an FPV Drone and, in particular, are used to compute spare points which will give us the actual orientation, in homogeneous coordinates, of the drone. We consider these points perturbed by some Gaussian noise.
Since the roto-translations are non linear functions, we can't handle those operations on the same unit Quaternion, so we must adopt a unit Quaternion for the trans-

lations, which involves linear operations, and another one for rotations performed with angles.

Other related works have studied tracking problem and kinematics of UAVs. An example is given in [26] in which is proposed a Visual Odometry system estimator based on the UKF. In this work the UKF is extend to handle representation of Lie Group and Lie Algebra. In [3] has been studied an algorithm to refine UAV coordinates, by binding the ground reference point. In this work, only a 2D model was considered, by maintaining the altitude a constant value.
We found our inspiration in [20] in which are described the applications of Kalman filter for the real-time estimation of a rigid body orientation from measurements of acceleration, angular velocity and magnetic field strength. Here is described the use of UKF based on Quaternion for estimating rigid body attitude, underlying that even if UKF handle non linear models, it needs some extensions to deal with state vectors that are not elements of a vector space.

1.3 Thesis Contents

After having described topics involving this work, let's describe our project. Some studies are similar to what we're going to analyze, but it seems that no one has conducted a comparison between different type of Kalman Filters and, moreover,there aren't application of the UKF filtering method with Unit Dual Quaternion for trajectory estimation of an UAV model. The first part, after the introduction, will present all most important kinds of filters invented: Linear Kalman Filter, Extended Kalman Filter, and Unscented Kalman Filter. The latter will be treated in detail by explaining how can be adapted to work with Dual Quaternions and its relation with Riemannian geometry.
Later on, a description and a background on Quaternions and related mathematical structures (Riemannian manifolds) will be explained, from the basics to the main features which are adopted to develop this thesis.
Those two parts will be followed by a methodology chapter, in which will be explained the backgrounds of the experimental phase, i.e. the object under examination, the FPV Drone, data collected and finally the behaviours of Linear Kalman filter, Unscented Kalman filter and Unscented Kalman filter with Dual Quaternions examined after the computation of our retrieved data.
Then a comparison between them will be argued, by showing numerical results and plotted graphs.
The thesis will end up with a conclusive summary about the main results obtained and the more effective tool for those kind of data; at the end are cited some future related works.

2 Filtering method: Kalman filters

2.1 Background

In this section we will introduce the basics of filtering methodology by describing the main problem and present most important algorithms.

2.1.1 Filtering problem

Generally speaking, when we are dealing with a physical system, which is driven by a set of external or internal components, like sensors, for metrics evaluations we would like to predict future possible behaviours in order to reduce the errors during the time.
In this context sensor errors and system errors are evaluated on the observation of the process, and it's require to find an optimization of the given parameters.

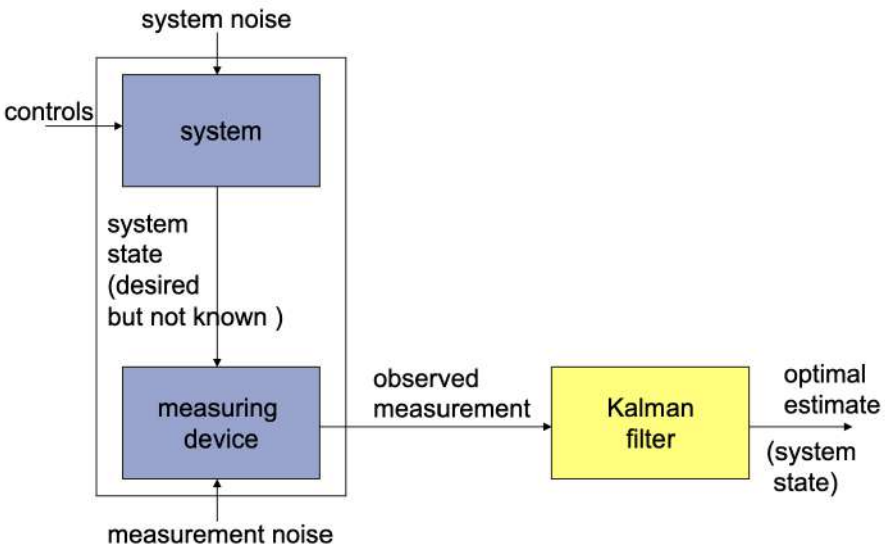

Figure 2.1: Typical application of Kalman Filtering

The Filtering problem can be formulated as follow, as cited in [40]:

$$\begin{aligned} x(k+1) &= f(x(k), u(k), w(k)) \\ y(k) &= h(x(k), v(k)) \end{aligned} \tag{2.1}$$

Where:

- $x \in \mathbb{R}^n$ is the system state vector.
- $f(\dots)$ corresponds to the system dynamics.
- $u \in \mathbb{R}^m$ is the control vector.
- w is the vector representing the system errors.
- y is the observation vector.
- $h(\dots)$ corresponds to measurement function.
- v corresponds vector that represents the measurement error sources.

Given all these parameters with a proper initialization, the main goal consists on finding the best estimate for $x(k)$, the system state vector at time k.
In general, any kind of filter attempts to extract an optimal approximation of the desired values (the system's state) from noisy data. In certain ways, the idea of optimality conveyed by the terms best estimate relates to the minimising of the state estimation error.

To handle with this many different approaches can be used:

- Mean, i.e., the center of the probability mass, corresponding to the minimum mean-square error.
- The mode that corresponds to the value of x that has the highest probability, corresponding to the Maximum a Posterior (MAP).
- The median, where the estimate is the value of x such that half the probability weight lies to the left and half to the right of it.

Under certain conditions related to the linearity of the system's dynamics (state and observation) and the normality of the random vectors involved (e.g., initial condition, state, and measurement noise), the conditional probability density functions propagated by the filter are Gaussian for every k. Thus, the mean vector and covariance matrix completely characterise the relevant PDF (Probability Distribution Function). Rather than propagating the whole PDF, the filter simply propagates (recursively) the first and second moments of the conditional PDF. The Kalman filter dynamics will be developed as a broad random parameter vector estimate. The Kalman filter calculates the minimal mean-square error estimate of the random vector that represents the system's state.

2.1.2 Relationship with Bayesian estimation

From a Bayesian perspective [29], the filter propagates the conditional probability density function of the desired values, based on knowledge of the actual data from the measuring devices, i.e., the filter assesses and propagates the conditional PDF for increasing values of k:

$$p(x(k) \mid y(1), \ldots, y(k), u(0), \ldots, u(k-1)) \tag{2.2}$$

The hidden state is defined as unobserved *Markov process* in recursive Bayesian estimation, and the measurements are the observed states of a hidden Markov model. When estimating the state x with a Kalman filter, the probability distribution of interest is the one associated with the current states based on measurements up to the current timestep. This is performed by removing the previous states and dividing by the probability of the measurement set. As a result, the predict and update steps of the Kalman filter based their computations on the probabilities that affect this variables. The probability distribution associated with the predicted state is the sum (integral) of the products of the probability distribution associated with the transition from the $(k-1)^{th}$ to the $(k)^{th}$ timestep and the probability distribution associated with the previous state across all possible timesteps.

2.1.3 Gaussian Distribution

The normal or Gaussian distribution, is one of the most important distribution in statistics. In general, most of the problems involving large amount of data can be solved by using this type of distribution. it's a continuous distribution, but it is also used to approximate discrete one.

In general there are lot of Normal distributions, and it's possible to fix ours by adjusting its mean μ and the variance σ^2. Now suppose X to be a random variable that has this normal distribution, so we can write $X \sim N(\mu, \sigma^2)$ which means " X is normally distributed with mean μ and variance σ^2." Where μ and σ^2 are the parameters of the normal distribution.
Looking at Figure [2.2] we can see that increasing μ shifts the normal density to the right without changing its shape. Increasing σ^2 flattens the density without shifting it.

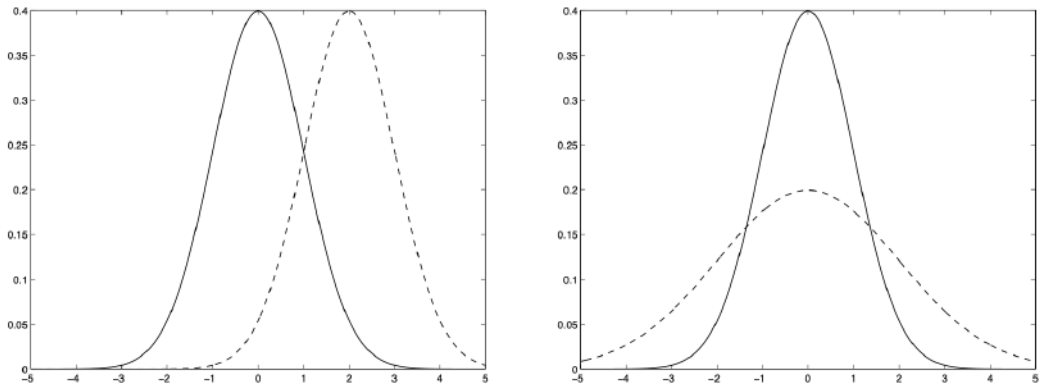

Figure 2.2: left figure shows two normal distributions with different means ($\mu = 0$ solid line, $\mu = 2$ dashed line) and same standard deviation ($\sigma = 1$). Right figure shows two normal distributions with the same mean ($\mu = 0$) and different standard deviations ($\sigma = 1$ solid line, $\sigma = 2$ dashed line)

The general form of a Normal distribution's Probability Density Function (PDF) is the following :

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{2.3}$$

So, as we said before, primary sources of a Kalman filter are assumed to be independent Gaussian's random variables with 0 mean, and so formally [33]:

A continuous random variable Z is said to be a standard normal (standard Gaussian) random variable, shown as $Z \sim N(0,1)$, if its PDF is given by

$$f_Z(z) = \frac{1}{\sqrt{2\pi}} e^{\left\{-\frac{z^2}{2}\right\}}, \quad \text{for all } z \in \mathbb{R} \tag{2.4}$$

After having described the Gaussian distribution and his PDF, we must remember that an important assumption on Kalman filter is that each data analyzed is perturbed by noise which is following the Gaussian distribution.

2.2 Linear Kalman Filter

As told in [18]: "if a dynamic system is linear and with Gaussian noise, the optimal estimator of the hidden states is the Kalman Filter".
In the Engineering scene, Kalman Filters are perhaps the most well-known model to reduce noise from sensor signals. As we will find, these models are very powerful on the approximation of data which are generally Gaussian distributed.

The optimality of Kalman filter in fact assumes that errors have Normal (Gaussian) distribution. Moreover it's based on the assumption that data under examination are modelled as a discrete time random variables, that means that are not directly observable but their measurement can be inferred by a sequence of noisy measurements $y1, y2, y3$.[17]

2.2.1 The Algorithm

As told in previous sections, the main algorithm is divided in 2 parts :

1. *Predict state* : the computation the new state is based on previous state and its related covariance
2. *Update state* : based on re-computation of the new covariance, the Kalman gain (trust of prediction) and update the new state

Predict state:

$$\hat{\mathbf{x}}_k = F_{k-1}\mathbf{x}_{k-1} \tag{2.5}$$

$$\hat{\mathbf{P}}_k = F_{k-1}\mathbf{P}_{k-1}F_{k-1}^T + \mathbf{Q}_{k-1} \tag{2.6}$$

Update step :

$$\tilde{\mathbf{y}}_k = \mathbf{y}_k - H_k\hat{\mathbf{x}}_k \tag{2.7}$$

$$\mathbf{S}_k = H_k\mathbf{P}_kH_k^T + \mathbf{R}_k \tag{2.8}$$

$$\mathbf{K}_k = \mathbf{P}_kH_k^T\mathbf{S}_k^{-1} \tag{2.9}$$

$$\mathbf{x}_k = \hat{\mathbf{x}}_k + \mathbf{K}_k\tilde{\mathbf{y}}_k \tag{2.10}$$

$$\mathbf{P}_k = \hat{\mathbf{P}}_k - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T \tag{2.11}$$

Moreover, it's important to specify that this algorithm works on the basis of two important models : the *System Model* and the *Measurement Model*. The first model, is represented from this equation :

$$\mathbf{x}_k = F_{k-1}\mathbf{x}_{k-1} + \mathbf{v}_{k-1} \quad \mathbf{v}_{k-1} \sim N(0, \mathbf{Q}_{k-1}) \tag{2.12}$$

Where :

- x_k is a state vector at time k
- F_{k-1} is the state transition matrix (the model is linear). The matrix can change during the time
- v_{k-1} is the vector that account the noise in the system model (i.e. the model is an approximation of the real one)
- Q_{k-1} describe the noise PDF defined by it's covariance matrix.

The *System model* relates the state vector at time x_k to x_{k+1}.
The multiplication obtained by F_{k-1} (state transition matrix at previous frame) and x_{k-1} is linear.
Usually F_{k-1} is stable but can change during the time interval, and the same can happen to the covariance matrix.
Usually the latter is difficult to determine, and one of the best practice is to guess it. We start to compute this covariance matrix with a diagonal matrix and then we see if it fits the original model.

The *Measurement model* instead is composed by this elements:

$$\mathbf{y}_k = H_k\mathbf{x}_k + \mu_k \quad \mu_k \sim N(0, \mathbf{R}_k) \tag{2.13}$$

in which :

- H_k is the matrix which maps the current state x_k to the measurements.
- μ_k is the vector that accounts for noise in measurements.
- R_k measurement noise PDF is fully defined by its covariance matrix.

Let's now analyze the algorithm in detail.
We assume to have an initial state, normally distributed with covariance P_0 mean x_0.

The equation (2.5) shows the estimate of the hidden state given the previous hidden state. F is the transition Matrix from the previous state estimation to the current state estimation.

The equation (2.6) is the updated estimation of the uncertainty P. This is the covariance matrix, and it is a function of the covariance matrix of the previous state.

In (2.7) $\hat{y}_k$ is the measurement residual and it is obtained once we have the predicted state $\hat{x}_k$, and multiplied it by H_k (we obtain the *expected measurement*), and then subtract y_k which is the actual measurement from the expected measurement.

In (2.8) we get the measurements of residual covariance.

In (2.9) we get Kalman gain with the use of residual covariance. The Kalman gain K at a timestep n is given by the previous estimation of the uncertainty P, the linear function H, and the inverse of the new covariance S_k^{-1}.The gain is used to give a weight to the new values, and say if data computed are reliable.

Finally (2.10) and (2.11) describe the new state and the new covariance computed.

The Kalman filter(KF) is one of the most widely used methods for tracking and estimation due to its simplicity, optimality, tractability and robustness. However, the application of the KF to nonlinear systems can be difficult.
Let's see how this problem can be solved.

2.3 Extended Kalman Filter

As already said, KF have problems on handling Non Linear systems.
In general, but also as said here [15], "the most common approach is to use the Extended Kalman Filter (EKF) which simply linearize all nonlinear models so that the traditional linear Kalman filter can be applied."
Although the EKF (in its many forms) is a widely used filtering strategy, it is difficult to implement, difficult to tune, and reliable for systems which are almost linear on the time scale of the update intervals.
Moreover it has been discovered that it is not an optimal estimator, in fact it presents two important drawbacks:

- The Jacobian matrices are nontrivial to estimate and this lead to significant implementation difficulties in most applications.
- The Linearization process can produce highly unstable filters if the assumptions of local linearity is violated.

Now, let's understand how it linearize non linear models : for this purpose, it will be more clear with the use of two pictures.

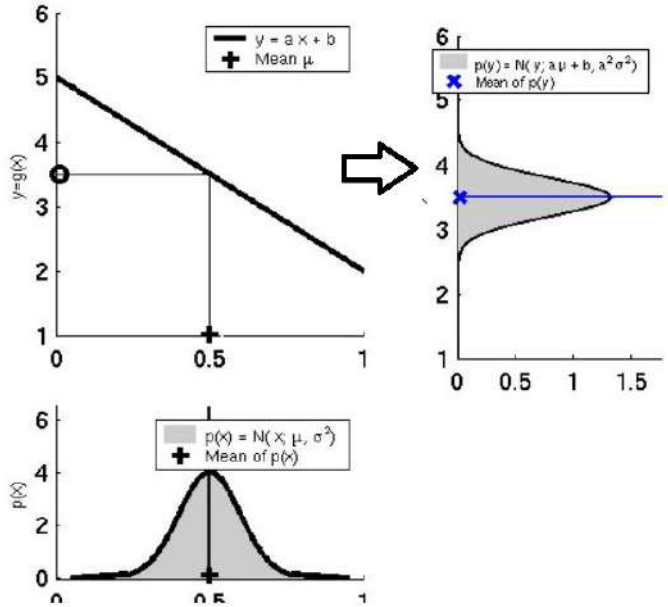

Figure 2.3: Linear transformation of a Gaussian

Figure [2.3] shows the effect of a linear transformation of a Gaussian random variable. That was the case of the original Kalman filter.
The lower right graph shows the random variable distributed about its mean. It is transformed through the linear function ($y = ax + b$) represented by the upper left graph. The resulting distribution (upper right) is again Gaussian and it is correct.

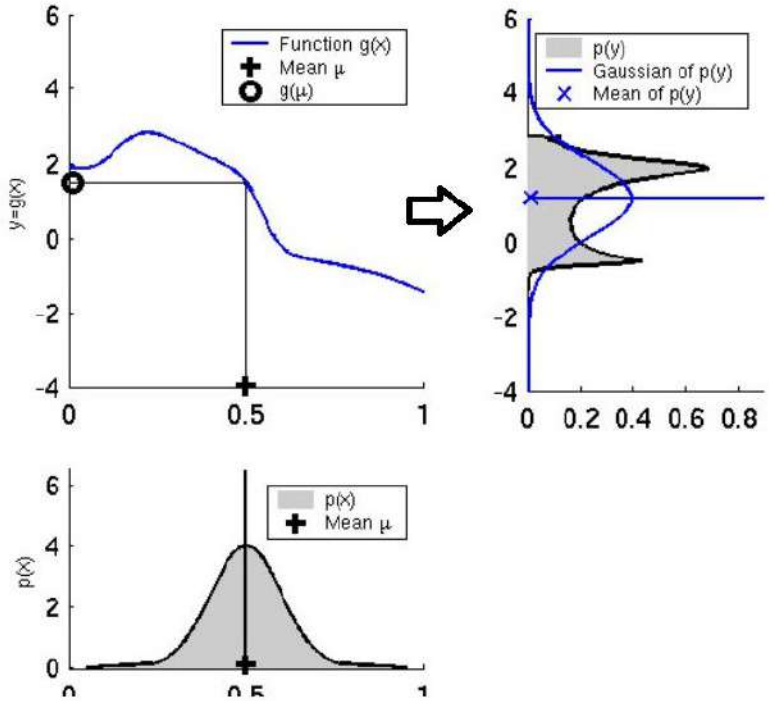

Figure 2.4: Non Linear transformation of a Gaussian

Instead, if we apply a non linear function to a Gaussian curve, the result of the transformation won't be a Gaussian anymore. The figure [2.4] shows how this phenomena can happened.
As said before, we look at the image from the bottom left to the top right. Following this path we can see that in the image at the top left of the image we have a non-linear function that "mirrors" the Gaussian one. Since the curve is not linear and visibly deformed, the resulting transformation (top right) will take on another shape and will not be Gaussian.
The EKF linearizes non linear models in this way: uses the Taylor series expansion to approximate the transformations using the first order of its series expansion evaluated at the mean estimate of x at time t.

2.3.1 Taylor's Expansion and EKF

The Taylor's expansions series, in mathematics, is represented by a function which involves in a infinite sum of terms that are calculated from the values of it's derivatives at a single point.
In practice we can say that in Taylor series a point is taken and are performed bunch of derivative on it.
When a Taylor series is performed for a polynomial then the result is also a polynomial and, accordingly with this, what EKF does is that it first evaluate the non-linear function to a mean which is the best approximate of the distribution and then es-

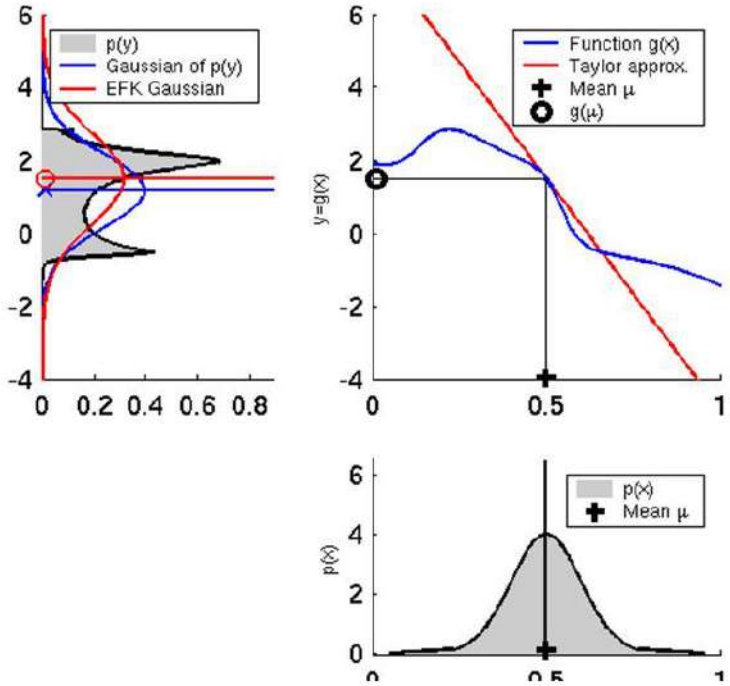

Figure 2.5: EKF Linearization

timate a line whose slope is around that mean. This slope is determined by the first order derivative of the Taylor expansion as the first order derivative gives a linear value. Therefore this method that allows to retrieve the slope from the first derivative of the Taylor series is know as First order Taylor expansion.
The figure shown above 2.5 describe in visible way what is described so far.
The Gaussian's shown in the bottom picture is transformed by using the non linear function $g(x)$ and the resulting shape, on the top left picture, is obviously a non Gaussian function.
By considering the tangent line passing through the non-linear curve and a little neighborhood of points on that, is possible to get a good approximation of the Gaussian. So the colored line (red) corresponding to the Taylor approximation considers only points of the non linear curve on the tangent neighborhood, and hence, it's possible to conclude that the function in that point is linear.

2.3.2 EKF Algorithm

The state transition and observation models in the extended Kalman filter needn't to be linear functions of the state; alternatively, they might be differentiable functions.

$$\boldsymbol{x}_k = f\left(\boldsymbol{x}_{k-1}\right) + \boldsymbol{w}_k \tag{2.14}$$

$$\boldsymbol{y}_k = h\left(\boldsymbol{x}_k\right) + \boldsymbol{v}_k \tag{2.15}$$

Both process model and measurement model are perturbed by process and observation noises, $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$, that are respectively considered to be zero mean multivariate Gaussian noises with covariance Q_k and R_k.

The function h is used to compute the predicted measurement from the predicted state, while the function f is used to compute the anticipated state from the prior estimate.
f and h, on the other hand, cannot be directly applied to the covariance.
Instead, a Jacobian matrix of partial derivatives is computed. The Jacobian is evaluated with current anticipated states at each time step. The Kalman filter equations can be employed with these matrices. The non-linear function is effectively linearized around the present estimate in this process.

Predict state:

$$\hat{\mathbf{x}}_k = f(\mathbf{x}_{k-1}) \tag{2.16}$$

$$\hat{\mathbf{P}}_k = Fj_{k-1}\mathbf{P}_{k-1}Fj_{k-1}^T + \mathbf{Q}_{k-1} \tag{2.17}$$

Update step :

$$\tilde{\mathbf{y}}_k = \mathbf{y}_k - h(\hat{\mathbf{x}}_k) \tag{2.18}$$

$$\mathbf{S}_k = H_j\mathbf{P}_k H_j^T + \mathbf{R}_k \tag{2.19}$$

$$\mathbf{K}_k = \mathbf{P}_k H_j^T \mathbf{S}_k^{-1} \tag{2.20}$$

$$\mathbf{x}_k = \hat{\mathbf{x}}_k + \mathbf{K}_k\tilde{\mathbf{y}}_k \tag{2.21}$$

$$\mathbf{P}_k = \hat{\mathbf{P}}_k - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T \tag{2.22}$$

The main difference between the Linear kalman filter are :

- The F_{k-1} matrix will be replaced by Fj_{k-1} when calculating $\hat{\mathbf{P}}_k$
- The H matrix in the Kalman filter will be replaced by the Jacobian matrix H_j when calculating $\mathbf{S}_k$, $\mathbf{K}_k$, and $\mathbf{P}_k$
- To calculate $\tilde{\mathbf{y}}_k$, the h function is used instead of the H matrix
- To calculate $\hat{\mathbf{x}}_k$ the f function is used instead of the F matrix
- Both of above function described are not linear

2.3.3 Drawbacks on EKF

By linearizing state and/or measurement equations and applying the usual Kalman filter formulae to the resultant linear estimation issue, the extended Kalman filter solves the nonlinear estimation problem.
The linearization produces approximation mistakes, which the filter ignores during the prediction and update processes. As a result, the error estimates of the Extended Kalman filter tend to underestimate state uncertainty.

In fact, as mentioned in [15], since the Extended Kalman filter uses the Taylor's expansion series to approximate the mean and the distribution of a random variable mapped by a non-linear function f, it was proved that these approximations are accurate only if the second and higher order terms in the mean and higher order terms in the covariance are negligible. But unfortunately this is not the case, and in many practical situations linearization introduces significant biases or errors.

2.4 The Unscented Kalman Filter

2.4.1 Improvements of EKF

The Unscented Kalman filter (UKF) is a nonlinear process and measurement model extension of the traditional Kalman filter.
The fundamental distinction between the UKF and the well-known Extended Kalman Filter (EKF) is that the UKF approximates the Gaussian probability distribution with a collection of single points (called *Sigma Points)*, whereas the EKF linearizes the (nonlinear) model equations. Because the original equations are employed, the results are typically more accurate and less expensive to compute (no jacobian matrices need to be computed).

2.4.2 The unscented Transformation

When we talk about UKF is important to delineate the main differences and improvements with respect to the EKF.
As we have seen before, When a Gaussian is processed by a non linear function, it no longer stays a Gaussian; The new filter is based on a newer technique to compute the approximation of a Gaussian distribution after a non linear transformation. This technique is called *Unscented Transformation* and it is based on a computation of new approximation's points, on the neighborhood of the Gaussian distribution, called *Sigma Points*.

To summarize, the unscented transform goes through the following steps:

- Compute Set of Sigma Points.
- Assign Weights for each computed Sigma Point.
- To get the set of changed sigma points in the state space, process each point through the function f.
- Compute Gaussian from weighted and transformed points.
- Compute Mean and Variance of the new Gaussian.

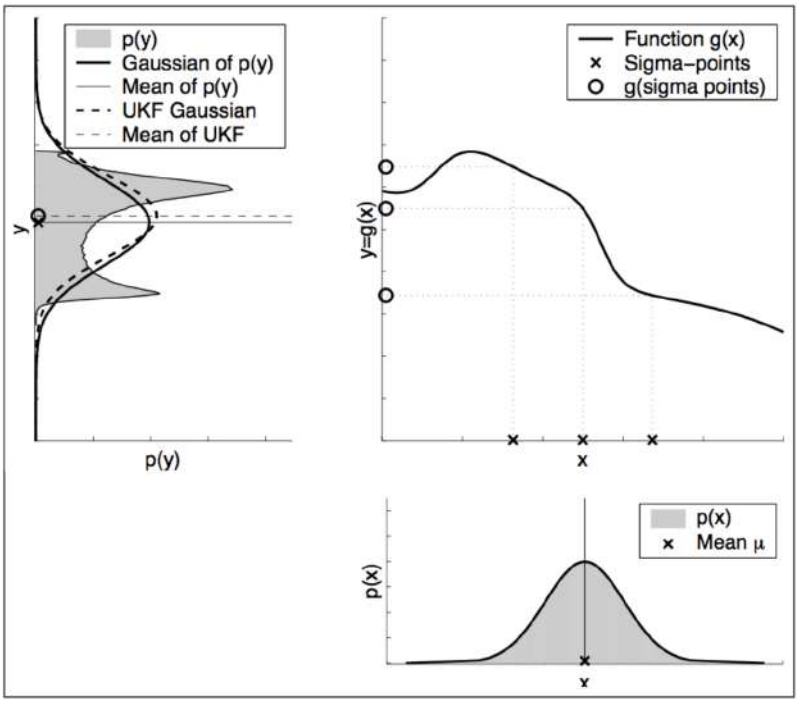

Figure 2.6: UKF transform with the use of Sigma Points

Sigma Points are computed following the formula written below, as the result of the research made in [15]:

$$\begin{array}{llll} \mathcal{X}_0 & = \overline{\mathbf{x}} & W_0 & = \kappa/(n+\kappa) \\ \mathcal{X}_i & = \overline{\mathbf{x}} + \left(\sqrt{(n+\kappa)\mathbf{P}_{xx}}\right)_i & W_i & = 1/2(n+\kappa) \\ \mathcal{X}_{i+n} & = \overline{\mathbf{x}} - \left(\sqrt{(n+\kappa)\mathbf{P}_{xx}}\right)_i & W_{i+n} & = 1/2(n+\kappa) \end{array} \tag{2.23}$$

in which :

- The Sigma Point Matrix is denoted by $\mathcal{X}$. It's important to remember that each column of $\mathcal{X}$ represents a group of sigma points. So, if we're working in two dimensions, the matrix size of $\mathcal{X}$ will be 2×5 because each dimension has a total of five sigma points.
- κ is the scaling factor that determines how far from the mean our sigma points should be chosen. According to a thorough mathematical analysis, the best number for κ is 3-n.
- W_i is the weight associated with the i^{th} point
- $\mathbf{P}_{xx}$ and $\overline{\mathbf{x}}$ are respectively the Covariance and the mean associated with the i^{th} point

- The script $\sqrt{(n+\kappa)\mathbf{P}_{xx}}$ deals with the Square root of a matrix, and it is defined when a matrix S satisfies the condition: the matrix under the square root need to be positive semi-definite (i.e. if and only if $A = B^T B$ for some matrix B)

Regarding the computation of the weights of *Sigma Points* the equation are :

$$\begin{aligned} w^{[0]} &= \frac{\kappa}{n+\kappa} \\ w^{[i]} &= \frac{1}{2(n+\kappa)} \quad \text{for } i = 1, \ldots, 2n \end{aligned} \tag{2.24}$$

Finally after that, it is possible to recover the Gaussian that has been processed through the non linear function thanks to another bunch of equations :

$$\begin{aligned} \boldsymbol{\mathcal{Y}}^{[i]} &= f(\boldsymbol{\mathcal{X}}^{[i]}) \\ \bar{\mathbf{x}} &= \sum_{i=0}^{2n} w^{[i]} \boldsymbol{\mathcal{Y}}^{[i]} \\ \bar{\mathbf{P}} &= \sum_{i=0}^{2n} w^{[i]} \left(\boldsymbol{\mathcal{Y}}^{[i]} - \mu'\right) \left(\boldsymbol{\mathcal{Y}}^{[i]} - \mu'\right)^T \end{aligned} \tag{2.25}$$

For whose:

- $\bar{\mathbf{x}}$ corresponds to the predicted mean
- $\bar{\mathbf{P}}$ is the Predicted Covariance
- $w^{[i]}$ Weights at the corresponding on sigma Point.
- $\boldsymbol{\mathcal{X}}^{[i]}$ Sigma Point.
- $\boldsymbol{\mathcal{Y}}^{[i]}$ new sigma points transformed by function f.
- n is the reference dimension.

So that's all about the Unscented transformation and how it works. Now let's see how this implementation can be inserted in the main algorithm of the Kalman Filter.

2.4.3 The UKF algorithm

In this subsection will be presented the algorithm composed by the two common steps : The *Predict* and the *Update step*
The first one, is simply the application of what is written above, i.e. the Unscented Transformation.
Since we need to take into account the process noise, the equation (2.25) becomes:

$$\begin{aligned} \bar{\mathbf{x}} &= \sum_{i=0}^{2n} w^{[i]} \boldsymbol{\mathcal{Y}}^{[i]} \\ \mathbf{P}' &= \sum_{i=0}^{2n} w^{[i]} \left(\boldsymbol{\mathcal{Y}}^{[i]} - \mu'\right) \left(\boldsymbol{\mathcal{Y}}^{[i]} - \mu'\right)^T + \mathbf{Q}_t \end{aligned} \tag{2.26}$$

Where $\mathbf{Q}_t$ is the process noise.

What concerns the update step is similar to the original Kalman Filter and the fundamental step is convert the predicted state to the measurement state. For each new update we need to recompute Sigma points because they depend on the Mean and Variance computed at each update phase.
Now, since we have already computed the mean and the covariance, we need to instantiate each of the prediction points on the measurement model. To do so, the following equations are required:

$$\begin{aligned} \boldsymbol{\mathcal{Z}}^{[i]} &= h(\mathcal{Y}^{[i]}) \\ \boldsymbol{\mu}_z &= \sum_{i=0}^{2n} w^{[i]} \boldsymbol{\mathcal{Z}}^{[i]} \\ \mathbf{P}_z &= \sum_{i=0}^{2n} w^{[i]} \left(\boldsymbol{\mathcal{Z}}^{[i]} - \boldsymbol{\mu}_z\right) \left(\boldsymbol{\mathcal{Z}}^{[i]} - \boldsymbol{\mu}_z\right)^T + \mathbf{R} \\ y &= \mathbf{z} - \boldsymbol{\mu_z} \end{aligned} \tag{2.27}$$

Where :

- $\boldsymbol{\mathcal{Z}}^{[i]}$ are the new sigma points in the measurement space.
- $\boldsymbol{\mu}_z$ is the new Mean calculated.
- $\mathbf{P}_z$ is the new Covariance.
- h is the function which maps sigma points to measurements space.
- y corresponds the the residual of the measurements.

Since the algorithm doesn't linearize the Gaussian function there has been some changes while computing the Kalman Gain.
The more important step is to compute the cross-correlation between sigma points in state space and sigma points in measurement space in order to evaluate error in prediction.

$$\begin{aligned} T &= \sum_{i=0}^{2n} w^{[i]} \left(\boldsymbol{\mathcal{X}}^{[i]} - \bar{\mathbf{x}}\right) \left(\boldsymbol{\mathcal{Z}}^{[i]} - \boldsymbol{\mu}_z\right)^T \\ \mathbf{K} &= \text{ T. } \mathbf{P}_z^{-1} \end{aligned} \tag{2.28}$$

T corresponds to the cross-correlation matrix between the state space and the measurement space, $\mathbf{P}_z^{-1}$ to the Covariance Matrix and $\mathbf{K}$ is the Kalman Gain.

Finally we can update Covariance and Mean with new measurements, so the following finals equations will do that:

$$\begin{aligned} \mathbf{x} &= \bar{\mathbf{x}} + \mathbf{K}y \\ \mathbf{P} &= \bar{\mathbf{P}} - \mathbf{K}\mathbf{P}_\mathbf{z}\mathbf{K}^\mathbf{T} \end{aligned} \tag{2.29}$$

Even if the equations and the algorithm looks very different from previous EKF, it's quite similar. In fact the part which has made more changes on that was the presence of Sigma Points and the transformations of their values from the state space to the measurement space.
The comparison between Kalman Filter algorithm and Unscented one is shown below.

Kalman Filter	Unscented Kalman Filter
	$\boldsymbol{\mathcal{Y}} = f(\boldsymbol{\mathcal{X}})$
$\bar{\mathbf{x}} = \mathbf{F}\mathbf{x}$	$\bar{\mathbf{x}} = \sum w^m \boldsymbol{\mathcal{Y}}$
$\bar{\mathbf{P}} = \mathbf{F}\mathbf{P}\mathbf{F}^\mathsf{T} + \mathbf{Q}$	$\bar{\mathbf{P}} = \sum w^c (\boldsymbol{\mathcal{Y}} - \bar{\mathbf{x}})(\boldsymbol{\mathcal{Y}} - \bar{\mathbf{x}})^\mathsf{T} + \mathbf{Q}$
	$\boldsymbol{\mathcal{Z}} = h(\boldsymbol{\mathcal{Y}})$
	$\boldsymbol{\mu}_z = \sum w^m \boldsymbol{\mathcal{Z}}$
$\mathbf{y} = \mathbf{z} - \mathbf{H}\mathbf{x}$	$\mathbf{y} = \mathbf{z} - \boldsymbol{\mu}_z$
$\mathbf{S} = \mathbf{H}\bar{\mathbf{P}}\mathbf{H}^\mathsf{T} + \mathbf{R}$	$\mathbf{P}_z = \sum w^c (\boldsymbol{\mathcal{Z}} - \boldsymbol{\mu}_z)(\boldsymbol{\mathcal{Z}} - \boldsymbol{\mu}_z)^\mathsf{T} + \mathbf{R}$
$\mathbf{K} = \bar{\mathbf{P}}\mathbf{H}^\mathsf{T}\mathbf{S}^{-1}$	$\mathbf{K} = \left[\sum w^c (\boldsymbol{\mathcal{Y}} - \bar{\mathbf{x}})(\boldsymbol{\mathcal{Z}} - \boldsymbol{\mu}_z)^\mathsf{T}\right]\mathbf{P}_z^{-1}$
$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{K}\mathbf{y}$	$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{K}\mathbf{y}$
$\mathbf{P} = (\mathbf{I} - \mathbf{K}\mathbf{H})\bar{\mathbf{P}}$	$\mathbf{P} = \bar{\mathbf{P}} - \mathbf{K}\mathbf{P}_\mathbf{z}\mathbf{K}^\mathsf{T}$

Figure 2.7: KF algorithm vs UKF

3 Quaternions and Dual Quaternions

In this chapter we are going to explain the basics of Quaternions and Dual Quaternions and their role on mathematical theory.

3.1 Theoretical background

When we talk about Quaternions, we must consider Complex Numbers.
The Complex Number system adds a new set of numbers called imaginary numbers in addition to the well-known number sets (Natural, Integer, Real, and Rational). Imaginary numbers were created to solve problems for which there were no solutions, such as for the following equation:

$$x^2 + 1 = 0 \tag{3.1}$$

If we want to solve the equation we must state that $x^2 = -1$ which implies that there are not solutions since the square of any number is a positive number.
Imaginary number where invented to give a plausible solution at those type of equations which resolves in an impossible solution.

The imaginary number has the following form :

$$i^2 = -1 \tag{3.2}$$

The set in which all those numbers are clustered is the set $\mathbb{I}$ which is a subset of Complex numbers set $\mathbb{C}$, which includes also real numbers $\mathbb{R}$.
In the following we recall the most important operations that are possible with complex numbers:

- The **addition** and **subtraction** of Complex numbers is performed by adding or subtracting the real part to the imaginary part:

$$\textbf{Addition}\,(a_1 + b_1 i) + (a_2 + b_2 i) = (a_1 + a_2) + (b_1 + b_2)\, i \tag{3.3}$$

$$\textbf{Subtraction}\,(a_1 + b_1 i) - (a_2 + b_2 i) = (a_1 - a_2) + (b_1 - b_2)\, i \tag{3.4}$$

- **Multiplication** of a Complex number by a Scalar:

$$\lambda(a + bi) = \lambda a + \lambda bi \tag{3.5}$$

- **Product** of Complex numbers:

$$\begin{aligned} z_1 &= (a_1 + b_1 i) \\ z_2 &= (a_2 + b_2 i) \\ z_1 z_2 &= (a_1 + b_1 i)\,(a_2 + b_2 i) \\ &= a_1 a_2 + a_1 b_2 i + b_1 a_2 i + b_1 b_2 i^2 \\ &= (a_1 a_2 - b_1 b_2) + (a_1 b_2 + b_1 a_2)\, i \end{aligned} \tag{3.6}$$

- **Square** of Complex numbers:

$$\begin{aligned} z &= (a + bi) \\ z^2 &= (a + bi)(a + bi) \\ &= \left(a^2 - b^2\right) + 2abi \end{aligned} \tag{3.7}$$

- The **Conjugate** of Complex number is represented with the letter z^* and has the particularity to have the imaginary part negated:

$$\begin{aligned} z &= (a + bi) \\ z* &= (a - bi) \end{aligned} \tag{3.8}$$

- **Powers** of the imaginary part:

$$\begin{aligned} i^0 &= && 1 \\ i^1 &= && i \\ i^2 &= && -1 \\ i^3 &= ii^2 && = -i \\ i^4 &= i^2i^2 && = 1 \\ i^5 &= ii^4 && = i \\ i^6 &= ii^5 = i^2 && = -1 \end{aligned} \tag{3.9}$$

This last operation described is useful to understand the Complex Plane 3.1:

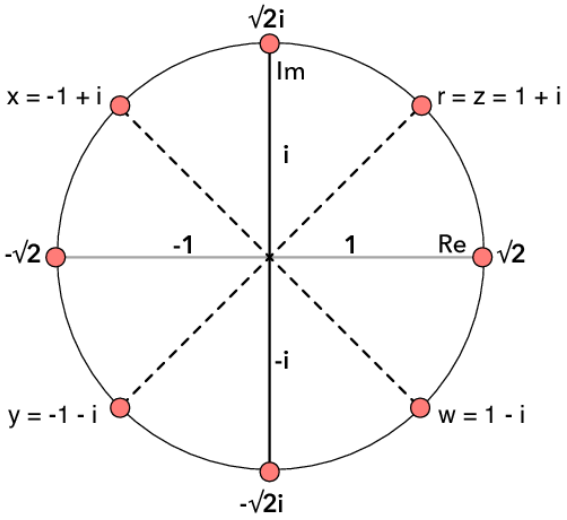

Figure 3.1: Complex plane and rotations

The idea is that also Complex numbers can be represented on a plane. The x axis belongs to real coordinates, instead the imaginary part is on the corresponding y axis.
In fact, as shown in the figure 3.1, if we multiply a number by **i** we get different rotations that can occur on that plane.

Rotations, in a complex plane, and with complex numbers, have the following formulation:

$$q = \cos\theta + i\sin\theta \tag{3.10}$$

The multiplication by q of any complex number is generalized below:

$$\begin{aligned} p &= a + bi \\ q &= \cos\theta + i\sin\theta \\ pq &= (a + bi)(\cos\theta + i\sin\theta) \\ a' + b'i &= a\cos\theta - b\sin\theta + (a\sin\theta + b\cos\theta)i \end{aligned} \tag{3.11}$$

which in matrix form has the following formulation:

$$\begin{bmatrix} a' & -b' \\ b' & a' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \tag{3.12}$$

After this brief introduction to the set of Complex Numbers $\mathbb{C}$, it's now possible to introduce Quaternions.

3.2 Unit Quaternion

The first approach to Quaternions was made in 1819 by W.R. Hamilton and the literal meaning that world is "a set of four".
Since complex numbers can have their own representation as points in planes, Hamilton wanted to do the same with points in 3D space. By walking on a street in Dublin on that years blow up in his mind the main formula, which represent the Quaternion multiplication, and revolutionised the algebra's world:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{3.13}$$

3.2.1 Basics

When we talk about Quaternions, in practice we are referring to an extension of our 3D space, that is computed by adding two more complex number in addition to **i**.
The general form of a Unit Quaternion is the following one:

$$q = s + xi + yj + zk \qquad s, x, y, z \in \mathbb{R} \tag{3.14}$$

where s, x, y and z are the real numbers and **i**, **j** and **k** are the basic Quaternions, which can be considered the unit-vectors of the three axes.
s is the the scalar part and $xi + yj + zk$ is the vector one.
Also Quaternions have their own set class, and it's formed by 4-dimensional vector space on real numbers, in which $\{1, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$ is a basis.

3.2.2 Main Operations and Properties

- **Addition** and **Subtraction** of Quaternions is computed in the following way:

$$\begin{aligned} q_a &= [s_a, \mathbf{a}] \\ q_b &= [s_b, \mathbf{b}] \\ q_a + q_b &= [s_a + s_b, \mathbf{a} + \mathbf{b}] \\ q_a - q_b &= [s_a - s_b, \mathbf{a} - \mathbf{b}] \end{aligned} \tag{3.15}$$

- **Product** of two Quaternions:

$$\begin{aligned} q_a &= [s_a, \mathbf{a}] \\ q_b &= [s_b, \mathbf{b}] \\ q_a q_b &= [s_a, \mathbf{a}]\,[s_b, \mathbf{b}] \\ &= (s_a + x_a i + y_a j + z_a k)\,(s_b + x_b i + y_b j + z_b k) \\ &= (s_a s_b - x_a x_b - y_a y_b - z_a z_b) \\ &\quad + (s_a x_b + s_b x_a + y_a z_b - y_b z_a)\, i \\ &\quad + (s_a y_b + s_b y_a + z_a x_b - z_b x_a)\, j \\ &\quad + (s_a z_b + s_b z_a + x_a y_b - x_b y_a)\, k \end{aligned} \tag{3.16}$$

  Which result is another Quaternion. This operation is Called Hamilton Product

- **Real Quaternion** and **Pure Quaternion** :
  By definition a Real Quaternion is a Quaternion with the characteristics of having vector term of 0:

$$q = [s, \mathbf{0}] \tag{3.17}$$

  Instead, a Pure Quaternion has the scalar part equal to 0

$$q = [\mathbf{0}, v] \tag{3.18}$$

- **Conjugate**:
  Similarly to complex number, the conjunction of a Quaternion corresponds to its own inverse, and by applying this operation twice we get back the original element. As a consequence, the conjugate of a product of two Quaternions is equal to the product of conjugates in reverse order:

$$(pq)^* = q^* p^* \text{and not } p^* q^* \tag{3.19}$$

- **Unit Quaternion**:
  Unit Quaternion has norm equal to 1.

$$||q|| = 1 \tag{3.20}$$

Unit Quaternion is so important because through it we can represent rotations of an angle θ, by a vector $\vec{v}$, in the euclidean 3-dimensional space:

$$\mathbf{q} = (\cos(\theta/2)), \vec{v}\sin(\theta/2)) \tag{3.21}$$

3.2.3 Quaternions for rotations

By following the guide in [5] is possible to find many different operations that allow us to recover Quaternion different structures, i.e. Matrices.
By recalling that a *pure Quaternion*, a Quaternion having no real part $q = 0 + xi + yj + zk$, can be represented as a vector in 3d space.
A Quaternion q_r can represent a rotation if its norm $|q_r|$ is equal to 1.
The operation that allows a rotation frame a coordinate of frame A to a coordinate of frame B is the conjugation.

$$q_B = q_R\, q_A\, q_R^* \tag{3.22}$$

The new Quaternion q_B is also a vector since it doesn't contain the real part:

$$\begin{aligned} q_R\, q_A\, q_R^* &= (q_0 + q_1 i + q_2 j + q_3 k)\,(xi + yj + zk)\,(q_0 - q_1 i - q_2 j - q_3 k) \\ &= \left(x\left(q_0^2 + q_1^2 - q_2^2 - q_3^2\right) + 2y\,(q_1 q_2 - q_0 q_3) + 2z\,(q_0 q_2 + q_1 q_3)\right) i + \\ &\quad \left(2x\,(q_0 q_3 + q_1 q_2) + y\left(q_0^2 - q_1^2 + q_2^2 - q_3^2\right) + 2z\,(q_2 q_3 - q_0 q_1)\right) j + \\ &\quad \left(2x\,(q_1 q_3 - q_0 q_2) + 2y\,(q_0 q_1 + q_2 q_3) + z\left(q_0^2 - q_1^2 - q_2^2 + q_3^2\right)\right) k \end{aligned} \tag{3.23}$$

This computation is also free from trigonometric functions, that means that only addition and multiplication appear on this formula.

3.2.4 From a Quaternion to a rotation matrix

If we need to recover a Rotation Matrix from a Quaternion q we need to express the Multiplication $q_R q_A q_R^*$ as a matrix M:

$$M = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_0 q_2 + q_1 q_3) \\ 2(q_0 q_3 + q_1 q_2) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_0 q_1 + q_2 q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

Since the norm of a rotation Quaternion is equal to 1 it's possible to simplify the matrix as following:

$$|q| = q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \tag{3.24}$$

for which derives the following equations:

$$\begin{aligned} -q_2^2 - q_3^2 &= q_0^2 + q_1^2 - 1 \\ -q_1^2 - q_3^2 &= q_0^2 + q_2^2 - 1 \\ -q_1^2 - q_2^2 &= q_0^2 + q_3^2 - 1 \end{aligned}$$

After these operations the matrix becomes :

$$M = 2 \cdot \begin{bmatrix} q_0^2 + q_1^2 - 0.5 & q_1 q_2 - q_0 q_3 & q_0 q_2 + q_1 q_3 \\ q_0 q_3 + q_1 q_2 & q_0^2 + q_2^2 - 0.5 & q_2 q_3 - q_0 q_1 \\ q_1 q_3 - q_0 q_2 & q_0 q_1 + q_2 q_3 & q_0^2 + q_3^2 - 0.5 \end{bmatrix}$$

3.2.5 From rotation matrix to Quaternion

We can compute the Quaternion that represents the same rotation given a rotation matrix M. First, compute the trace, which is the sum of the elements on the matrix's major diagonal:

$$\begin{aligned} \text{Trace}(M) &= M_{11} + M_{22} + M_{33} \\ &= 2\left(3q_0^2 + q_1^2 + q_2^2 + q_3^2 - 1.5\right) \\ &= 2\left(3q_0^2 + \left(1 - q_0^2\right) - 1.5\right) \\ &= 2\left(2q_0^2 - 0.5\right) \\ &= 4q_0^2 - 1. \end{aligned} \tag{3.25}$$

and by solving the equation by q_0 we get :

$$|q_0| = \sqrt{\frac{\text{Trace}(M)+1}{4}}$$

Once obtained q_0 we can obtain q_1:

$$M_{11} = 2\left(q_0^2 + q_1^2 - 0.5\right) = 2\left(\frac{\text{Trace}(M)+1}{4} + q_1^2 - 0.5\right)$$

$$|q_1| = \sqrt{\frac{M_{11}}{2} + \frac{1-\text{Trace}(M)}{4}}$$

and in a similar way also q_2 and q_3 by considering M_{22} and M_{33}:

$$|q_2| = \sqrt{\frac{M_{22}}{2} + \frac{1-\text{Trace}(M)}{4}}$$

$$|q_3| = \sqrt{\frac{M_{33}}{2} + \frac{1-\text{Trace}(M)}{4}}$$

3.2.6 From Euler angles to Quaternion

Given a rotation by angles, is possible to compute the corresponding Quaternion. Considering a rotation γ around the axis z:

$$M_\gamma = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

From the formula written above is possible to get q_0, q_1, q_2 and q_3:

$$\begin{aligned} |q_0| &= \sqrt{\tfrac{2\cos\gamma+1+1}{4}} = \sqrt{\tfrac{1+\cos\gamma}{2}} = \cos\tfrac{\gamma}{2} \\ |q_1| = |q_2| &= \sqrt{\tfrac{\cos\gamma}{2} + \tfrac{1-2\cos\gamma+1}{4}} = 0, \\ |q_3| &= \sqrt{\tfrac{1}{2} + \tfrac{1-2\cos\gamma+1}{4}} = \sqrt{\tfrac{1-\cos\gamma}{2}} = \sin\tfrac{\gamma}{2} \end{aligned}$$

Therefore:

$$q_\gamma = \cos\frac{\gamma}{2} + k\ \sin\frac{\gamma}{2} \tag{3.26}$$

3.3 Dual numbers

Before talking about Dual Quaternions is important to have a brief introduction on what dual numbers are based on.
When we deal with dual number we need to know that they aren't too different from Complex numbers $\mathbb{C}$.
In practice, any dual number, is formed by the *dual part* a_ε and a *non dual part* a_0. ε represents the dual unit and it satisfies the constraint $\varepsilon^2 = 0$

$$\hat{z} = a_0 + a_\epsilon \epsilon \tag{3.27}$$

They are used in different applications and geometry space, depending on the purposes:

- as a **Gaussian** representation
- as a **Polar** representation
- as an **Exponential** representations

3.3.1 Properties and Operations

Some important features are reported from [19] that have demonstrated some lemmas for which we will only make a brief citation:

- The **dual conjugate** is quite similar to the one of complex numbers
- The **addition** and **subtraction** are defined in the same way:

$$\hat{a} + \hat{b} = (a + b) + \varepsilon\,(a_0 + b_0) \tag{3.28}$$

- The **Multiplication**:

$$\begin{aligned} \hat{a}\hat{b} &= (a + \varepsilon a_0)\,(b + \varepsilon b_0) \\ &= ab + \varepsilon\,(ab_0 + a_0 b) \end{aligned} \tag{3.29}$$

- **Taylor Series** and Dual Numbers:
  Taylor expansion series in general is a way to express a function as an infinite sums of derivatives, so we expand a function a by displacement factor h and we definite in this way:

$$f(a+h) = f(a) + \frac{f'(a)}{1!}h + \frac{f''(a)}{2!}h^2 + \cdots \tag{3.30}$$

  For Dual Numbers we define a similar function but instead of h as a displacement factor we have $h\varepsilon$:

$$f(a+h\varepsilon) = f(a) + \frac{f'(a)}{1!}h\varepsilon + \ldots 0 \tag{3.31}$$

  Observing that all the second-order and higher derivatives are nullified and so, the surprising result is that we haven't an approximation but an exact result. That turns out to be useful when there is the need to evaluate differential functions.

All the other operations are well described in [36].
Dual numbers are important for their application for describing derivatives generated by a certain program: this method is called *Automatic Differentiation*, which recall the method of Taylor expansion series.
In practice given a function (possibly a polynomial), it's possible to predict second order derivative while computing curve trajectory. The output corresponds to two different values, the former as real part, the latter as dual part:

$$f(a+\varepsilon) = f(a) + f'(a)\varepsilon \tag{3.32}$$

3.3.2 Dual Vectors

A Dual Vector is a specific kind of dual number whose real and dual parts are both vectors; in other words, a dual vector is comparable to a regular vector with the distinction that its components are dual numbers.
A unit dual vector is one of several possible representations of lines in space. The real part is the unit direction vector of a line, and the dual part is the line moment with respect to the origin of the line.
In this context a particular case of dual vectors are Plücker lines, which can be represented in the following way:

$$\hat{\mathbf{l}} = \mathbf{l} + \varepsilon \mathbf{m} \tag{3.33}$$

m is the line moment and is obtained by having cross product between the origin point p and the line vector l and defined as:

$$\mathbf{m} = \mathbf{p} \times \mathbf{l} \tag{3.34}$$

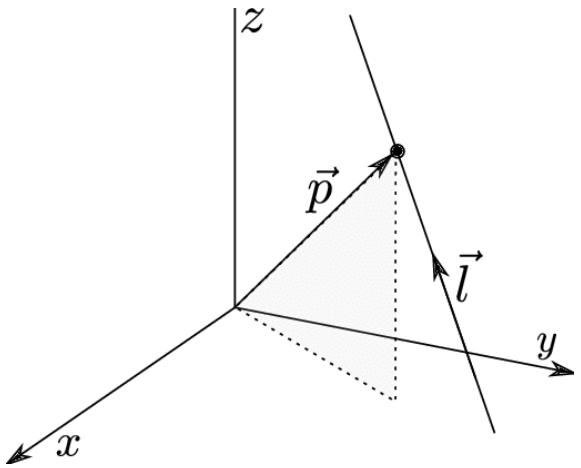

Figure 3.2: Geometrical representation of Plücker line

Thanks to Plücker Lines is possible to derive and prove the Dual Quaternion existence. For more detailed information see [2].

3.4 Dual Quaternions

Dual Quaternions are one of the most important element of this project because they allow us to unify the rotation and translation in an unique state and moreover represent a valid alternative to compute rigid motions.

3.4.1 Definition

The first approach of Clifford with Dual Quaternions, was confused about their possibilities. Dual Quaternions extend the above described Quaternion by one more unit.
It belongs to the the set $\mathbb{H}$ which is the set of the Hypercomplex numbers, in which are contained also dual numbers [50, 31].
A dual Quaternion can be written as :

$$\hat{\mathbf{q}} = \hat{w} + i\hat{x} + j\hat{y} + k\hat{z} \tag{3.35}$$

where :

- $\hat{w}$ is the dual number (scalar part)
- $(\hat{x}, \hat{y}, \hat{z})$ corresponds to the dual vector (vector part)
- i, j, k are the Quaternion units

3.4.2 Dual Quaternions Algebra

- **Addition**:

$$\begin{aligned}
\hat{A} =&(A, B) = a_0 + a_1 i + a_2 j + a_3 k + b_0\varepsilon + b_1\varepsilon i + b_2\varepsilon j + b_3\varepsilon k \\
\hat{C} =&(C, D) = c_0 + c_1 i + c_2 j + c_3 k + d_0\varepsilon + d_1\varepsilon i + d_2\varepsilon j + d_3\varepsilon k \\
\hat{A} + \hat{C} =&(A + C, B + D) = (a_0 + c_0) + (a_1 + c_1)\, i + (a_2 + c_2)\, j + (a_3 + c_3)\, k \\
&+ (b_0 + d_0)\, \varepsilon + (b_1 + d_1)\, \varepsilon i + (b_2 + d_2)\, \varepsilon j + (b_3 + d_3)\, \varepsilon k
\end{aligned} \tag{3.36}$$

Where, for simplicity reasons $\hat{A}$ is considered as an ordered pair of 2 Quaternions A and B. The summation is computed component wise.

- **Multiplication**:

$$\begin{aligned}\hat{A} =&(A, B) = A + \varepsilon B \\ \hat{C} =&(C, D) = C + \varepsilon \\ \hat{A}\hat{C} =&(A + \varepsilon B)(C + \varepsilon D) = AC + \varepsilon(AD + BC)\end{aligned} \tag{3.37}$$

  In this equation disappear the component BD because it's multiplied by ε^2 which is equal to 0

- **Conjugate** Dual Quaternions can have two different types of conjugate representation. The first one is simply the extension of conjugate of a normal Quaternion, which can be expressed in this way:

$$\hat{A}^* = (A^*, B^*) = A^* + \varepsilon B^* \tag{3.38}$$

  By recalling Dual numbers, the product of two Dual Quaternion conjugates, is the reversed order product of their conjugates:

$$\hat{G}^* = (\hat{A}\hat{C})^* = \hat{C}^*\hat{A}^* \tag{3.39}$$

  The other method consists on taking the dual number conjugate. The conjugate in this scenario is useful to get the mean for Quaternion conjugate or dual conjugate

- **Norm** of Dual Quaternion in general is computed to find their *magnitude*. The Norm is given by :

$$|\hat{A}| = \sqrt{\hat{A}\hat{A}^*} \tag{3.40}$$

  As concerning unit Quaternion, Dual Quaternions also can have unitary norm $|\hat{A}| = 1$ and are called Unit Dual Quaternions. Later we will see that they represent a Manifold, considered as a sphere, in which are permitted exponential and logarithm operations useful to mapping points and compute spatial Euclidean transformations.

- **Inverse** of Dual Quaternion : given $A + \varepsilon B$ being a Dual Quaternion where A is non 0, the inverse is computed as following

$$A^{-1}(1 - \varepsilon B A^{-1}) \tag{3.41}$$

3.4.3 Theory of Screw motion

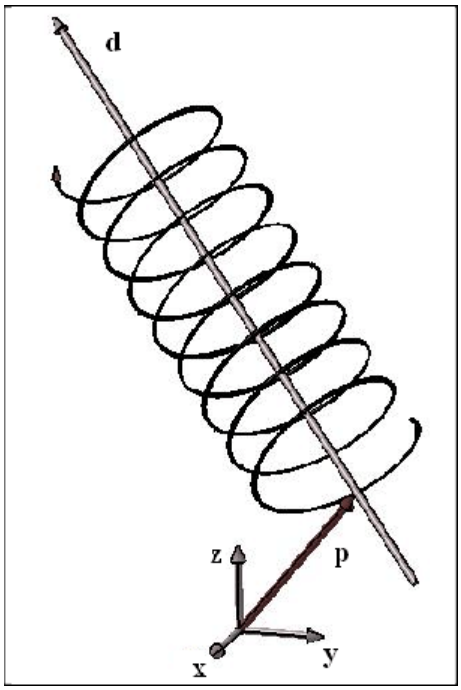

Figure 3.3: Screw Motion Geometry

The relation between Screw Theory, formulated by Sir Robert Ball, and Dual Quaternions has strong connection because, in the beginning, becomes an important instrument to operate with robots, body dynamics and computational geometry. The relation consist to compute rigid body motion, which is based on screw theory.

The screw motion theory is based on the famous Chasles' theorem, as shown below [3.4], which describes a screw displacement as a spatial displacement of a rigid body characterized by a rotation with angle θ about on axis at point $\mathbf{c}$ with direction $\mathbf{n}$ and a translation d parallel to the reference axis.
The four independent components of Plücker vector t that defines the screw axis n, coupled with the rotation angle about a linear slide down this line, forms a pair of vectors known as a screw. For comparison, the three Euler angles that define the rotation and the three components of the translation vector may also be used to determine the six parameters that describe a spatial displacement.

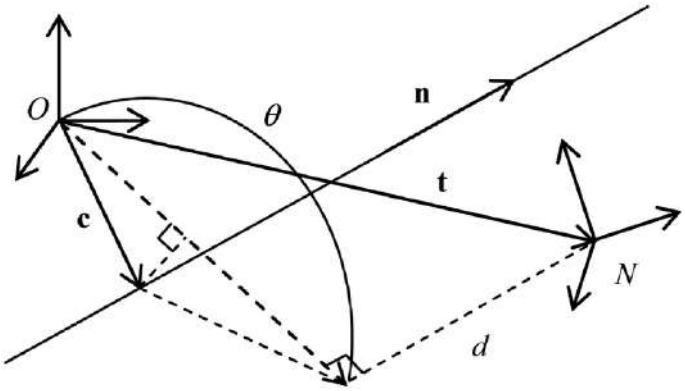

Figure 3.4: Chasles' theorem: translation and rotation is equivalent to a rotation about a screw axis together with a translation along the screw axis

3.4.4 Representation of Rotations and translation with Dual Quaternions

Unit Dual Quaternion can be explicitly written as a tuple representing the above described screw motion, defined as 6-DOF transformation consisting, as we have already said, on a rotation and a translation:

$$\mathbf{q} = \left[\cos\left(\frac{\hat{\theta}}{2}\right), \sin\left(\frac{\hat{\theta}}{2}\right)\hat{\mathbf{n}}\right] \tag{3.42}$$

$$\hat{\mathbf{q}} = \mathbf{q} + \frac{1}{2}\varepsilon\mathbf{q}\mathbf{t}_{ON} \tag{3.43}$$

Where :

- $\hat{\theta} = \theta + \varepsilon d$ is the dual angle corresponding to the screw axis $\hat{n}$
- $\mathbf{q} + \frac{1}{2}\varepsilon\mathbf{q}\mathbf{t}_{ON}$ is the kinematic equation of a screw displacement in compact form from point $\mathbf{t}_O$ to a point $\mathbf{t}_N$, represented by a dual Quaternion $\hat{\mathbf{q}}$.

The derivation of this formula can be studied on the appendix section A.3 of [19].

3.4.5 Relations with Manifold *SE(3)*

Dual Quaternion with unitary norm, i.e. Unit Dual Quaternions, can be identified in differentiable Riemannian manifold [7, 9]. It is 6-dimensional and it is embedded in 8-dimensional Euclidean space.
The manifold can be derived by this formula:

$$\Omega = \left\{ \left[\mathbf{x}_\mathrm{r}^\top, \mathbf{x}_\mathrm{d}^\top \right]^\top \mid \mathbf{x}_\mathrm{r} \in \mathbb{S}^3, \mathbf{x}_\mathrm{r}^\top \mathbf{x}_\mathrm{d} = 0 \right\} \subset \mathbb{R}^8 \tag{3.44}$$

where $\mathbf{x_r}$ is the real part representing a 3-DoF spatial rotation in the form of Quaternion in the hyper sphere $\mathbb{S}^3$, dot product his dual and orthogonal part $\mathbf{x_d}$ located in $\mathbb{R}^4$:

$$\mathbf{x}_\mathrm{r} = \left[\cos(\theta/2), \mathbf{n}^\top \sin(\theta/2) \right]^\top \in \mathbb{S}^3 \tag{3.45}$$

$$\mathbf{x}_\mathrm{d} = \frac{1}{2} \left[0, \mathbf{t}^\top \right]^\top \otimes \mathbf{x}_\mathrm{r} \in \mathbb{R}^4 \tag{3.46}$$

After having described how and where are located dual Quaternions $\mathbf{x_r}$ and $\mathbf{x_d}$, it's possible to describe possible operations that can be used to pass from the tangent space to the manifold one and vice versa:

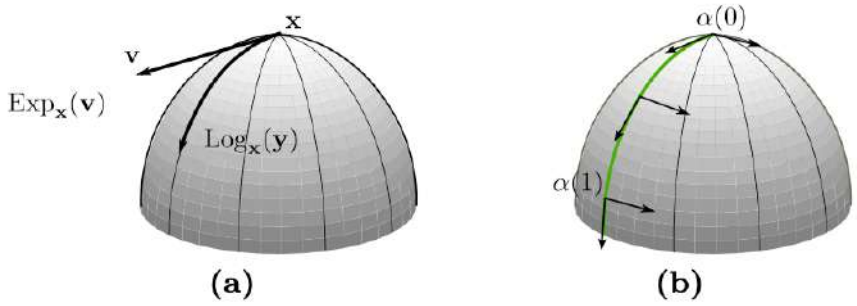

Figure 3.5: Graphical illustration of basic manifold operations. (a) The exponential and logarithm maps. (b) The parallel transport for moving vectors along a curve.

Defining Ω as our manifold, there exist, for Unit Dual Quaternions, an *Exponential Map* at a certain point $\mathbf{x} \in \Omega$ that maps a vector $\vec{v} \in T_x\Omega$ to the point $y = Exp_x(\vec{v}) \in \Omega$, by making possible that the curve l is a *geodesic* from x to y. A *Geodesic* is a curve representing the shortest path, in that case an arc, between two points on a Riemannian Manifold. A *Geodesically complete* Manifold is verified when the domain of a Geodesic curve can be extended to real numbers $\mathbb{R}$.

This kind of operation is in general allowed in a little neighborhood of the origin of the vector space $\in T_x\Omega$.
There exist the inverse operation, which permit to pass from the vector space to the manifold one, and it's called *Logarithm map* and denoted as $Log_x(y)$ and defined in neighborhood of $\mathbf{x}$.
Both the operations are shown in figure [3.6] below.

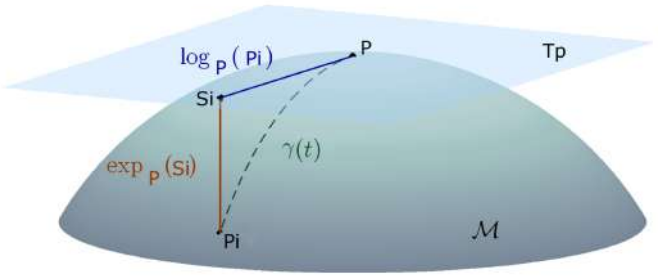

Figure 3.6: Tangent space of the manifold M at point P, Si the tangent vector of Pi and $\gamma(t)$ the geodesic between P and Pi.

Manifolds, despite those operations, are useful to evaluate statistics:

- If we consider a set of points $\mathbf{x} = x_1, \ldots, x_n$ we can compute the Empirical Mean by minimizing the sum of the squared distance:

$$\mathrm{E}[\mathbf{x}] = \arg\min_{\mu} \sum_{k=1}^{K} d^2\left(\mathbf{x}_k, \mu\right) \tag{3.47}$$

- If exists E[$\mathbf{x}$] we can compute the covariance by generalizing it in the following way:

$$\mathbf{P}_{\mu} = \frac{1}{K} \sum^{K} \log_{\mu}\left(\mathbf{x}_k\right) \log_{\mu}\left(\mathbf{x}_k\right)^T \tag{3.48}$$

Last but not least feature of manifolds is the capability of moving geometrical object along them by using a technique called *Parallel transportation along* α.
In practice given a curve α mapped from point 0 to 1 on the Manifold, formally speaking ($\alpha : [0,1] \rightarrow \Omega$), there exist an isometries which creates the Parallel transportation.
As mentioned in [12] "The parallel transport is the straightest, or least deforming way to move geometric objects along curves, and coupled with the exponential map, it provides the straightest way to move a geometric object from one point of the manifold to a neighbouring point via the geodesic curve that joins them."
In this Thesis, Unit Dual Quaternions applied with Unscented Kalman filter, represent a manifold, in which are possible to compute all the above operations in

order to have a better estimation of Mean and Covariance distribution during the computation.
The cited paper in fact, adopt a similar technique, by considering every sigma point σ generated a stochastic variable. Since Unit Dual Quaternions belong to a manifold, as we will see later we can use the above descripted techniques to compute efficiently mean and covariance, respectively up to the third order and second order.[15]

4 Methodology

In this section we will enter in details on how we have operated on our experiments. Firstly, we are going to introduce hardware and software utilized and then we pass trough of all step describing precisely the work done, i.e. data retrieval, model representation and how we have recovered the position of our drone in a different coordinate system.

4.1 Experimental tools

The experiments begin with setting up or FPV drone in order to recover useful informations for our tests.
FPV drones find their employment on cinematic recordings, video streams, sport environments. They are usually equipped with an High Definition Camera, different from that one mounted for real time video stream, in order to get detailed panoramic images that only drones are capable to reach off.

4.1.1 Hardware and Software

An FPV drone, consists of an UAV controlled by a pilot wearing smart googles receiving real time video feedback. FPV stands for *First Person View* and for this reason the pilot has the impression to be on board.
In general, drones can be controlled with different types of flight mode:

- VLOS stands for (*Visual line of Sight*) and indicates the visual flight condition, i.e. the one in which the drone is visible to the eye by the pilot who drives it remotely
- The (*Enhanced Visual Line Of Sight* (EVLOS) flying mode is an evolution of the VLOS flight mode. This term refers to a visual flight mode that involves subjects other than the pilot, such as a simple observer or "secondary" pilots, who are equipped with commands and in continual radio contact with the first pilot and can assume control of the drone if it disappears from their view.

- (*Beyond Visual Line Of Sight* (BVLOS) activities are those that are carried out at a distance that prevents the remote pilot from maintaining direct and continual visual contact with the aircraft, making it difficult to supervise the flight, maintain separations, and avoid crashes. FPV drones belongs to this category in which the pilot has no visual contact with the vehicle but uses tools (i.e. googles) that allow him to maintain control.

The image below shows flight modes [4.1].

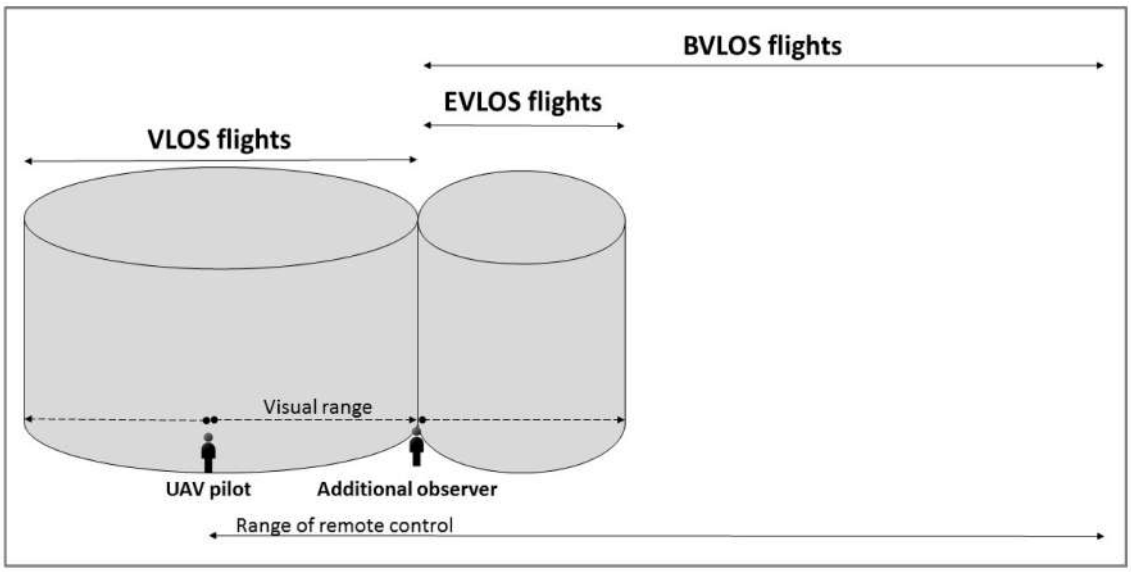

Figure 4.1: VLOS (*Visual line of Sight*), EVLOS (*Extended Visual line of Sight*) , BVLOS (*Beyond Visual line of Sight*)

The Drone used for tests was equipped the following components [4.2] :

- High resistance carbon frame.
- FC, *Flight controller* is the central unit of the drone, where are located sensors as accelerometer, gyroscope and barometer.
- *VTX*, the video transmitter unit, capable to transmit signal video to 5.8 Ghz band.
- *RX*, the receiver, which has the role of receiving commands from a controller called *TX*.
- *ESC* and *Motors*. Escs are the so called *Electronic speed controller*, and in practice handle power distribution in order to spin Motors faster or slower.
- an *FPV Camera* which capture the signal video from the local surrounding environment and send it to the VTX transmitter.
- *Propellers* and *Lipo Battery*.
- Additional *GPS Module* to recover drone in case of lost video or Rx signal.

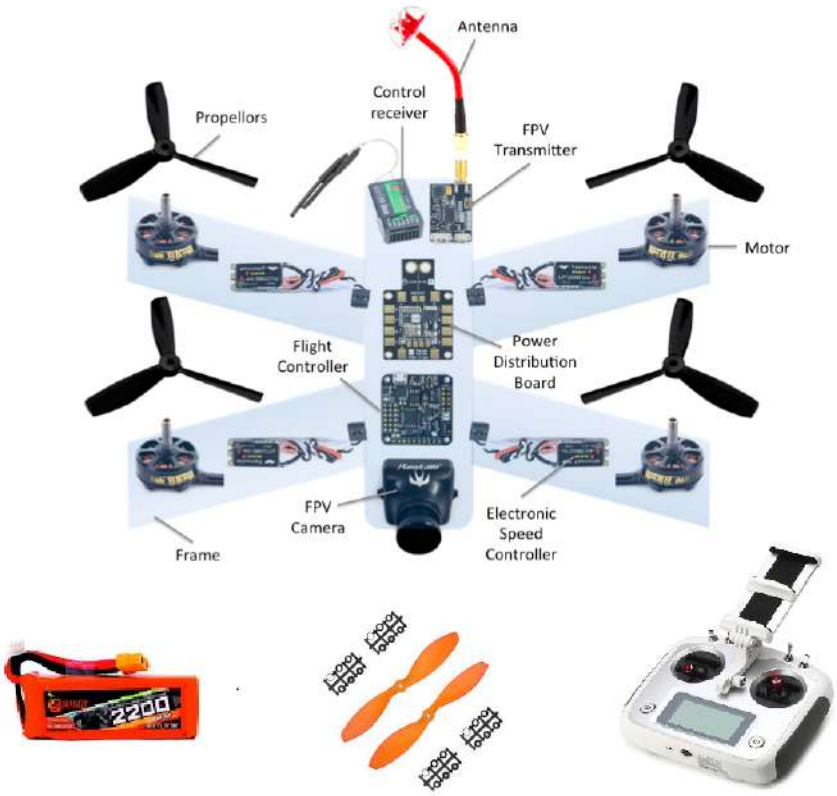

Figure 4.2: Drone components

Another important hardware component belonging to the FC board is the black box. It's a little memory unit already installed on the main board which allow us to record flight sessions, write retrieved data in a CSV file and finally discover, for example, if any sensor presents some problem.
This is particularly useful when, for example, we have Gyro which presents rumor noise on its data acquirement: in that case it's possible to smooth data using software helpers and, if the problem persists, the damaged unit must be replaced.
All data were saved in the black box, so we recovered them by using a parsing tool and converted in a readable CSV file.

Figure 4.3: Betaflight

What concerns the software, most of FPV drone runs Betaflight which is a free flight controller firmware used to fly multi-rotor craft and fixed wing craft. It's a fork, from Baseflight and Cleanflight, that improves flight performance, leading-edge feature additions, and wide target support.
It supports a wide range of flight controllers that have at least an STM32F4 Processor.
Betaflight Firmware supports the majority of remote control manufacturers such as FrSky, Graupner, Spektrum, DJI and FlySky. ESCs are controlled using a variety of available protocols including PWM, OneShot, MultiShot, DShot or even ProShot.
Even the less-related components do not remain untouched, with Betaflight allowing the control of many VTX and Camera settings directly from the flight controller.
By using Betaflight is possible to set up the drone with many different configurations.
For our purposes we set up the GPS and predisposed it, with IMU sensors, to log its data during the flight session.
Another important software tool, used to explore our data, is the *Betaflight Blackbox Explorer*

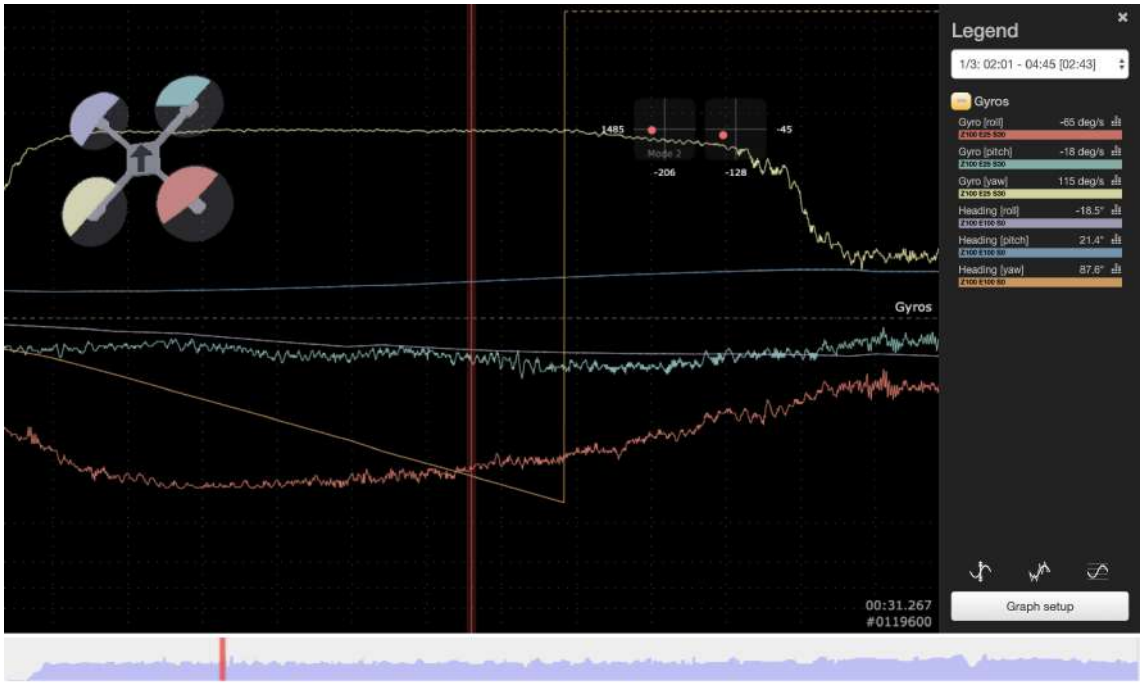

Figure 4.4: Betaflight Blackbox Explorer

It allowed us to export all data gained in a flight session and study them in a separate environment.
Anyway this is a definitive tool for analyze the spectrum created from data sensors, and visibly, is easy to evaluate a possible problem on hardware or software settings.

4.2 Model representation

Now we're going to introduce how we handled the UAV model in order to set it up for our tests. In particular, we considered angles and positions computed at each half-second during our flight session.

4.2.1 Recovering angles

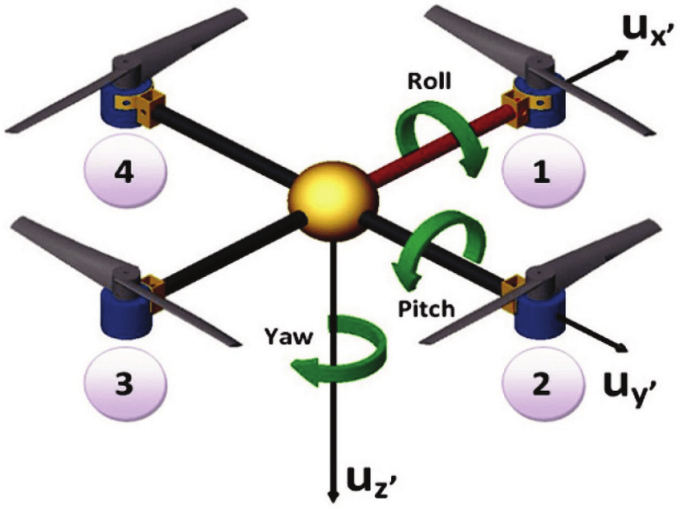

Figure 4.5: Model Representation

By referring on the above image [4.5], angles are retrieved from the drone's black-box in order to handle rotation of the rigid body.
Each angle is associated with a rotation on an axes. In particular:

- at the x axis corresponds the Roll movement (rotation that allows lateral movements of the vehicle). This kind of rotation varies from -180° to 180°.
- at the y axis corresponds the Pitch movement (rotation that allows Forward and Backwards movements of the vehicle) This kind of rotation varies from -180° to 180°.
- at the z axis corresponds the Yaw movement (rotation that allows Heading movements of the vehicle) This kind of rotation varies from 0° to 360°.

These angles will be used later to compute roto-translations by using Dual Quaternions.
The first three components of a Quaternion are in fact those angles, transformed in radians; instead the last component is a vector containing the actual position in a 3D euclidean space.

4.2.2 Recovering Position

How do we have computed movements? Positions and movements are calculated by manipulating data coming from the GPS [4.7]: we have considered the initial position referenced in Latitude and Longitude coordinates and, in a separate reference system that uses meters as unit measurement, we considered them at the stating point (0,0). In order to be able to compute that distance between two points, given their longitude and latitude, we used the *Haversine Formula*:

$$\text{hav}(\theta) = \text{hav}\left(\varphi_2 - \varphi_1\right) + \cos\left(\varphi_1\right)\cos\left(\varphi_2\right)\text{hav}\left(\lambda_2 - \lambda_1\right) \tag{4.1}$$

in which :

- φ_1, φ_2 are the latitude of point 1 and latitude of point 2.
- λ_1, λ_2 are the longitude of point 1 and longitude of point 2.

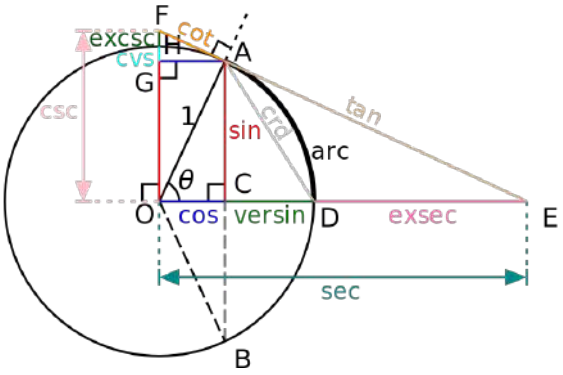

Figure 4.6: Geometrical representation of Haversine formulation

Pin Description:

PIN	PIN Name	I/O	Description
1	SDA	O	Compass SDA
2	GND	G	Ground
3	TX	O	Serial Data Output.
4	RX	I	Serial Data input.
5	VCC	I	3.0V~ 5.5V supply input,Typical: 5.0V
6	SCL	I	Compass SCL

Figure 4.7: Beitan BN-880 Gps

Since we were interested on finding the x, y and z coordinates (in meters) of the occurring displacement of our drone, the distance between the origin and the new point, is not sufficient. We needed to recover the intrinsic parameters of the equation of the line binding these two points.
The idea was to fix a starting point, in our case we fixed the initial position, given with a certain Latitude and Longitude, in a new 2-dimensional reference system at coordinate (0,0), i.e. the origin.
After that, for each new coordinate, we compute the distance from (0;0) of the new coordinates, by fixing one of the two coordinates each time, i.e. [0; y= new long coordinate] and [x= new lat coordinate; 0].
The below figure [4.8], represents this process. A different approach instead is used to compute the altitude, i.e. z statement of the vector.

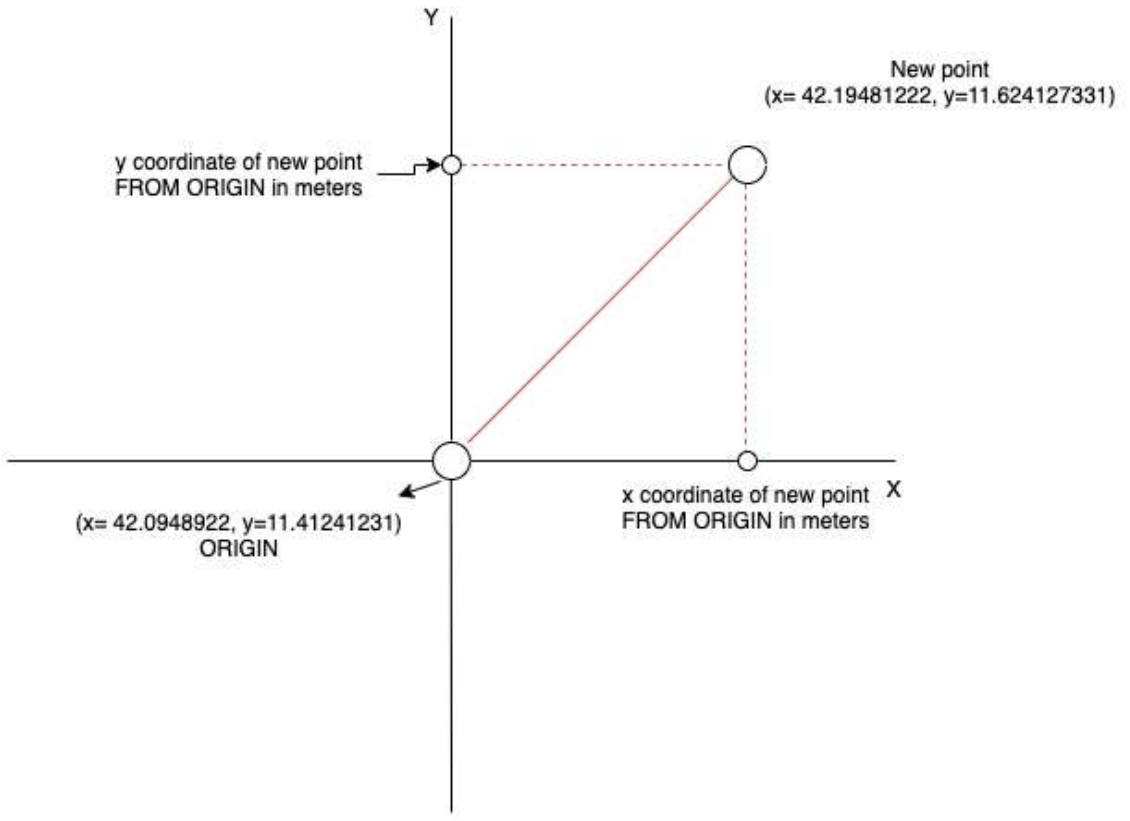

Figure 4.8: Computing intrinsic parameters of the straight line in red. Dotted lines in red are the distances computed by Haversine formula from that point to the axes

The z axis, that represents the altitude from the ground, is simply retrieved from the parsed data in meters, therefore we didn't need other computations.
After this description involving the preliminaries of the experiments, let's introduce the main work.

5 Tests and Results

5.1 Recorded session

Our test flight occurs in a parking spot, and the goal of the flight session was to perform a track in which it's possible to reach the case of warping angle.

Figure 5.1: View from the model: red circles represent GPS coordinates and number of satellites locked on

This kind of phenomena is reached when we are on the neighborhood of the 0° and 360° and it's attempted a left or right correction of the yaw axis (z). The problem is that every program is not able to automatically understand that, if we are at 0° oriented and we move slightly the yaw axis to the left, we reach with only 1 step the 360°: the computer will think that we passed from 0 to 360 with a complete rotation. To handle that we need pass through angles transformations, which involves non linear functions. This is the reason for which a normal Kalman filter can't handle that, since angles are treated as linear parts.

5.2 Experiment 1: Regular Kalman filter

The first attempt to verify that the prediction of filtering technique has been done correctly, was by applying the regular Kalman Filter to our retrieved data, in particular by observing angular changes during movements computed by the flying drone. The development environment chosen was the Python language, which offers very powerful 3d libraries to plot consistent graphs.

5.2.1 Initialization

Since the algorithm is recursive and needs to be updated at each iteration, we are going to focus on how we set up the *Covariance Matrix*, *State Transition Matrix*, *Hidden state* and relative *Noise* which perturbs measurements. On this experiment, the use of Dual Quaternions can't be applied since we are treating the problem as it has been a linear model. We used a matrix representation.

Recalling that, the *System model* to follow the equation 2.12 from Kalman Filtering section.

$$\mathbf{x}_k = F_{k-1}\mathbf{x}_{k-1} + \mathbf{v}_{k-1} \quad \mathbf{v}_{k-1} \sim N\left(0, \mathbf{Q}_{k-1}\right)$$

$$F_{k-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Hiddenstate = x = \begin{bmatrix} x \\ y \\ z \\ \alpha \\ \beta \\ \gamma \end{bmatrix}$$

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * 0.001$$

And recalling the *Measurement model* follow the equation 2.13:

$$\mathbf{y}_k = H_k \mathbf{x}_k + \mu_k \quad \mu_k \sim N(0, \mathbf{R}_k)$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * 100$$

Where :

- x_{k-1} is the *hidden state vector*, corresponding to the parameters at previous step needed to compute position and orientation of the actual step x.
- F_{k-1} is the initialized state transition matrix.
- Q_{k-1} Covariance matrix of the system model.
- H matrix mapping the current state to the measurements.
- R is the measurement noise covariance matrix. It's one of the components that will compose the *Kalman Gain*

Every experiment has the purposes to demonstrate that, given certain data perturbed by some Gaussian's noise, it's possible to handle it by well parametrizing the *Kalman gain*.
The Kalman gain is the amount of weight given to measurements and current-state estimates, and it can be "adjusted" to reach a specific result. With a large gain, the filter gives the most recent measurements more weight, and so conforms to them more quickly. The filter corresponds more closely to the model predictions when the gain is low. A high gain near to 1 will provide a more jumpy estimated trajectory, whereas a low gain close to 0 will level out noise but reduce responsiveness.

Since we considered our data perturbed not reliable at all, we have considered the above initialization data.

5.2.2 Plots and results Test 1

The result is shown below :

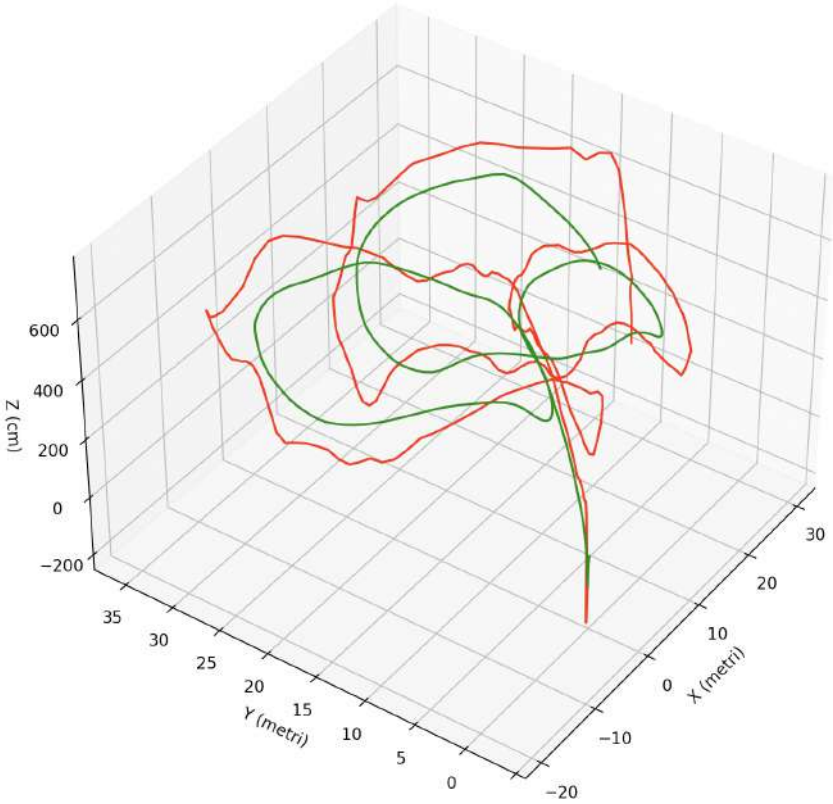

Figure 5.2: Plot of trajectory estimation given only GPS position data and Altitude from Barometer sensor: red line represent raw data, green line represent smoothed data from Kalman Filter.

As we can see in [5.2], since we were dealing with linear operations and we were not including velocities and angles, the experiment ended up with good results.

The trajectory estimated by Kalman Filter is smoother and more regular than the red one generated by raw data.
In the case in which we trust the measurements, it's possible to reduce the overall covariance, i.e reduce the Kalman gain, and we will see that green and red curves will be more next to each other.

Then we considered the case of the angles at each different frame.
The Graphical representation of the angles is given by a simple Cartesian 2d graph, in which in Y axis we considered the angle's amplitude and on X axis we considered the elapsed time in seconds.
The following figures will represent all different angles :

- Pitch α graph:

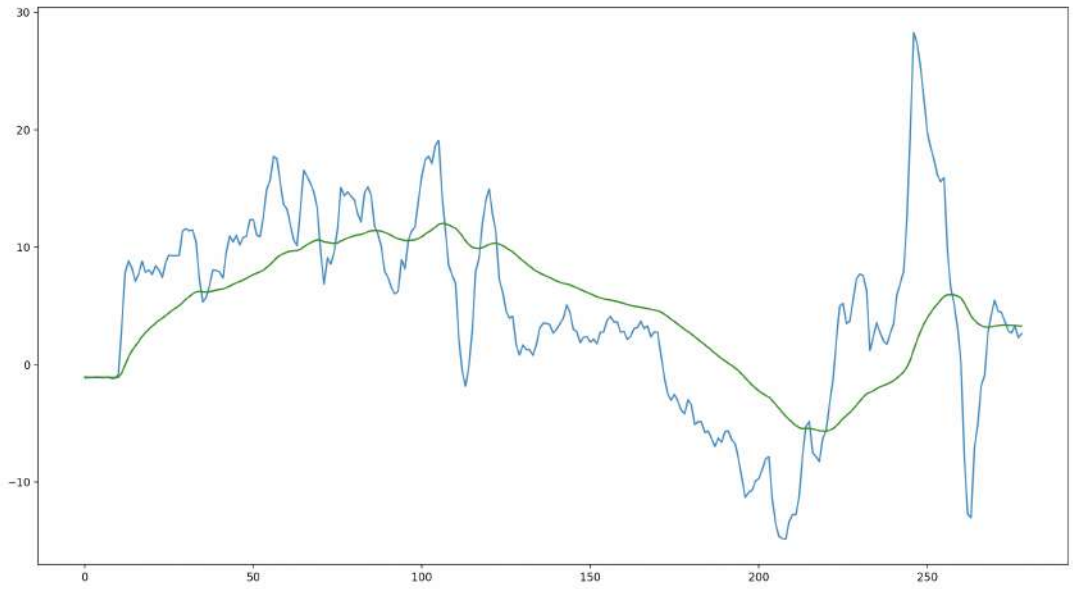

Figure 5.3: Pitch plot and Kalman Filter smooth

The graph shows that the prediction works well, and it's the same also for β angle. That is because there are no warping angles.

The situation changes with the angle Yaw shown in the next page.

- Roll β graph:

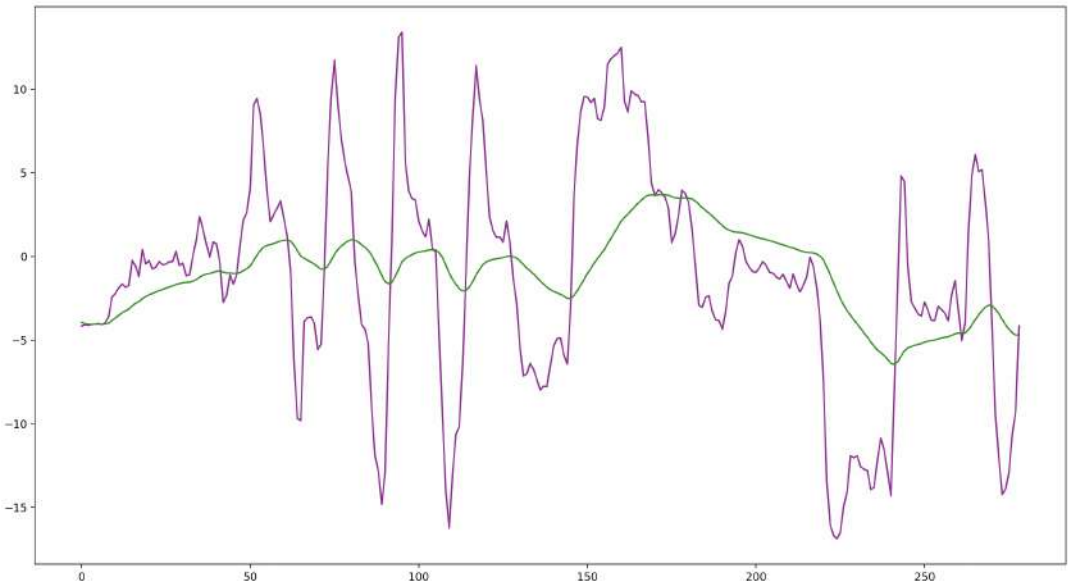

Figure 5.4: Roll plot and Kalman Filter smooth

- Yaw γ graph:

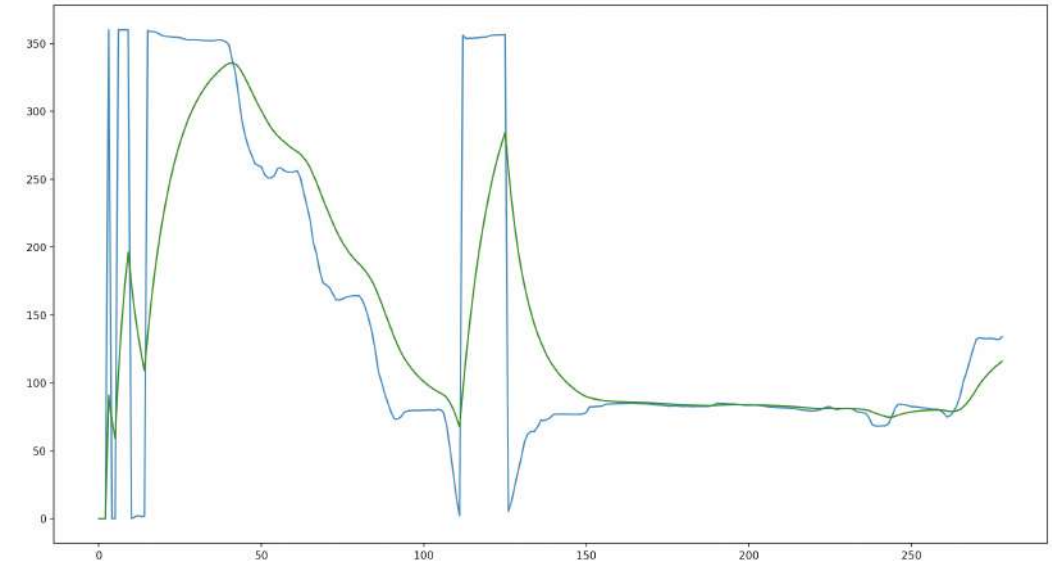

Figure 5.5: Yaw plot and Kalman Filter smooth

By looking at the last plot, we recognize that it's not correct.
There is a non-linearity that we are not considering, the fact that angles are modular. The residual is the difference between the measurement and the prior projected into measurement space. The angular difference between 359° and 1° is 2°, but 359° - 1° = 358°.

This observation is also valid for pitch and roll, but, since in our experiment we have had a variance of maximum 30 degrees, we never had a complete twist on the pitch and roll axis.
Instead this is completely verified at the yaw axis.
In the next experiment will be shown how this problem was handled by the UKF, which computes sums of weighted values in the unscented transform.

5.3 Test 2: Unscented Kalman filter

The second experiment consists on applying the same data used before, and see how the UKF resolve the non-linearity problems involving angles.
First of all, we considered the initialization of the corresponding parameters. Most of them were set the same as the previous experiment, just to have a more reliable comparison.

5.3.1 Initialization

The Unscented Kalman Filter takes in input the following parameters :

- The Observed parameters are 6, so the hidden state is a vector composed by the three canonical positions in 3d space, i.e. x, y and z, and the 3 angles corresponding the rotation of each axis: δ for *Pitch*, θ for *Roll* and γ for *Yaw*
- Process noise matrix: the process noise covariance per unit time, equal to a 6 by 6 matrix, has the same initialization to the previous used in regular Kalman filter.

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * 100$$

- Initial covariance matrix: the process noise covariance per unit time, equal to a 6 by 6 matrix, has the same initialization to the previous used in regular Kalman filter.

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * 0.001$$

- *Sigma Points*: sigma points generated for this experiment are equal to the the double of the number of state variables + 1.

  Sigma Points must be tuned with three parameters, accordingly with [46, 47, 45].

  The first parameter α determines the spread of sigma points, and typically is a small number (in our case 0.04). The second one is $\kappa = 3 - n$ or 0. The last β parameter, $\beta = 2$ is ideal for Gaussian distribution, as in our case.

- Recalling that in the Unscented Kalman Filter hasn't state transition matrices but operates with Sigma points passed through an identity function $f(\dots)$ and a non linear function $g(\dots)$ the following design have been implemented:

  - *System model* : this time the hidden state presents one more variable, since we have decomposed the γ angle in its *sine* and *cosine*, so it is defined as follows :

    $$x = \begin{bmatrix} x \\ y \\ z \\ \delta \\ \theta \\ \sin(\gamma) \\ \cos(\gamma) \end{bmatrix}$$

    and consequently the system model is defined as :

    $$x = f(x_{k-1}) + \mathcal{N}(0, Q) \tag{5.1}$$

– *measurement model*:

$$y = \begin{bmatrix} x \\ y \\ z \\ \delta \\ \theta \\ \gamma \end{bmatrix}$$

In order to handle modular angles such as *Yaw* the non linear function $g(\dots)$ converts the state x to an y state, in the measurement space, containing 1 less variable, and so by converting the sine and cosine in the correspondent angle γ .

This reasoning can be also applied to all the other angles, but for simplicity reasons, we have adopted this method only for γ angle.

After that, new sigma points are computed, and new mean and covariance are estimated.

The non linear function $g(\dots)$ is the *atan2* that given the sine and cosine of an angle gets back its value *Yaw* angle.

Formally speaking the *atan2* is defined as follow:

$$\text{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{if } x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{if } x < 0 \text{ and } y \geq 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{if } x < 0 \text{ and } y < 0 \\ +\frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0 \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0 \end{cases}$$

This kind of function as we can see, with respect to the simplest *arctan*, the sign of y and x arguments are handled.

The final measurement model equation is

$$y = g(x_k) + \mu_k \quad \mu_k \sim N(0, \mathbf{R}_k) \tag{5.2}$$

5.3.2 Plots and Results test 2

After having described the initialization of the Unscented Kalman filter, we have run our experiment and got feedback by plot visualization.
The first plot, by using almost the same parameters with respect to covariance and noise distribution, regards the motion of the drone.
In this case the Kalman gain has a smaller value, since the noise perturbing measurement model is not to strong. So we trusted the measurements already given.

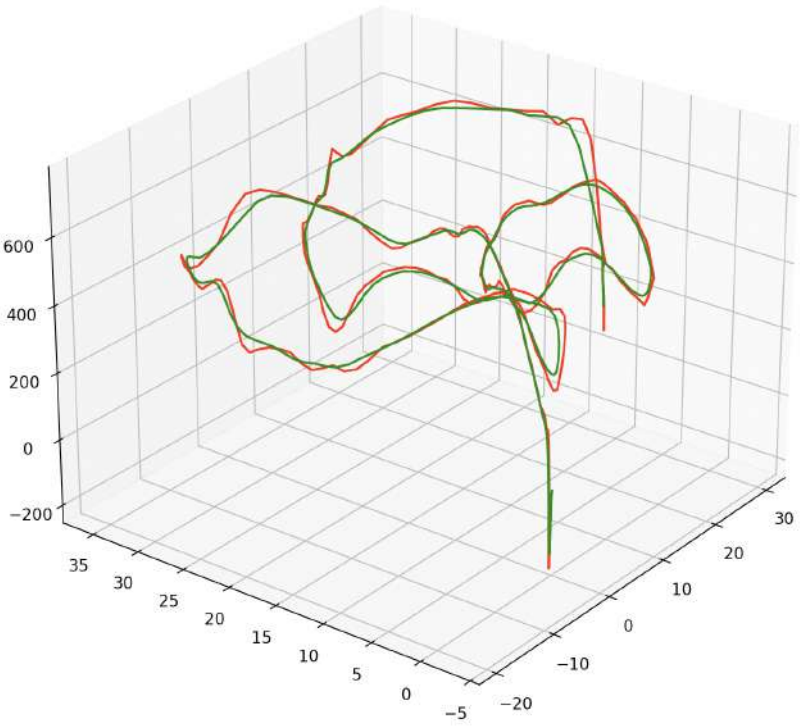

Figure 5.6: Plot of trajectory estimation given only GPS position data and Altitude from Barometer sensor: red line represent raw data, green line represent smoothed data from Unscented Kalman Filter.

The followings instead [5.7, 5.8, 5.9], are plots of angles: horizontal axis correspond to the time elapsed, instead vertical axis is the angle amplitude.

- Pitch δ :

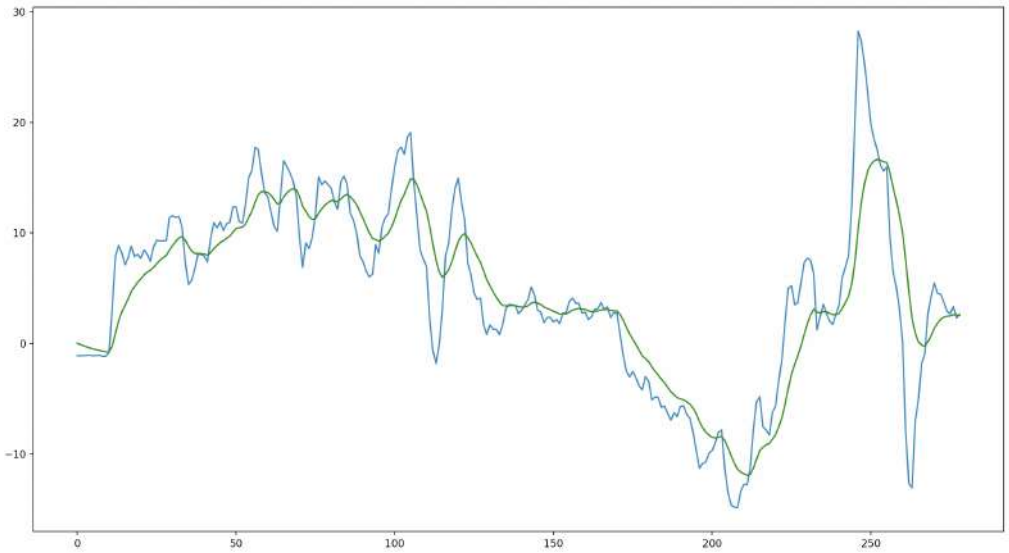

Figure 5.7: Plot of angle Pitch estimation : blue line represent raw data, green line represent smoothed data from Unscented Kalman Filter.

- Roll θ :

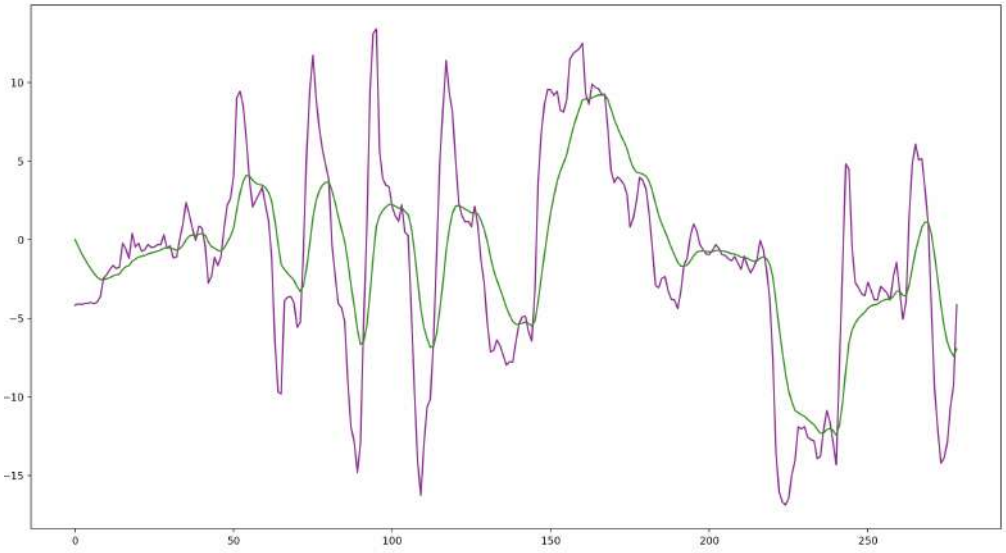

Figure 5.8: Plot of angle Roll estimation : purple line represent raw data, green line represent smoothed data from Unscented Kalman Filter.

No one of the graph's angles shown above has problems. The unscented Kalman filter works as well as the previous one in this case.

- Yaw γ :

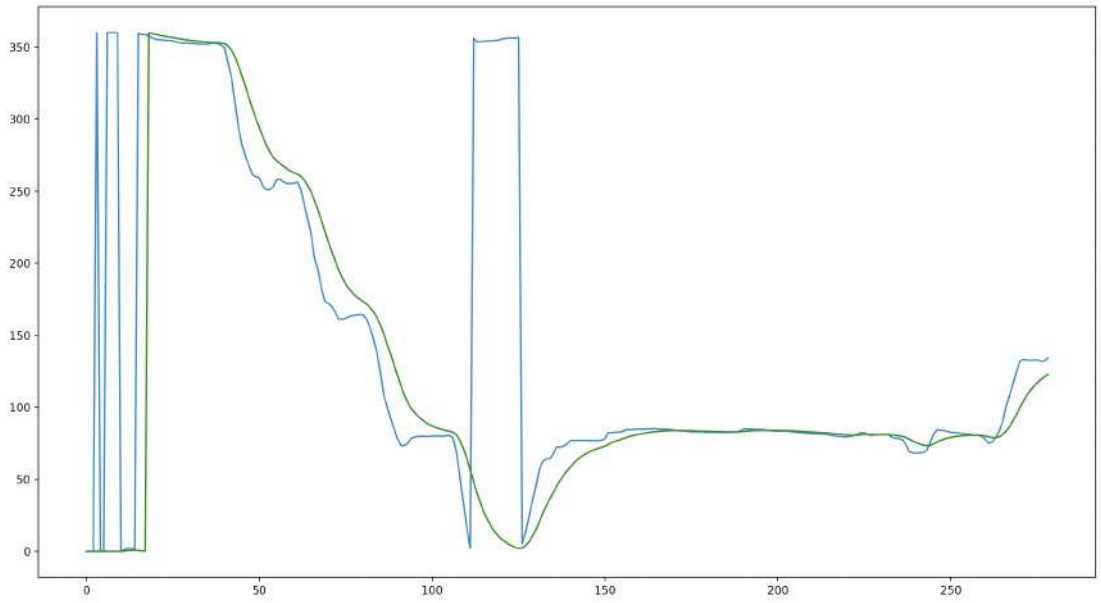

Figure 5.9: Plot of angle Yaw estimation : blue line represent raw data, green line represent smoothed data from Unscented Kalman Filter.

Figure [5.9] shows that the problem encountered with the regular Kalman Filter is vanished.
By looking at the middle of the graph, between the 100^{th} and 150^{th} seconds, is possible to notice that the green line hasn't followed the blue line, because, thanks to the non-linear function $g(\dots)$ that maps sigma points to a new measurement space, the Kalman Filter understands that the difference between 360° and 0° is equal to 0.

5.4 Experiment 3 : Unscented Kalman filter Dual Quaternions

The final test regards the application of the above cited Dual Quaternions to represent the rigid body motion in a unique state.
Even in this case we were considering as our hidden state a Dual Quaternion composed by a unit Quaternion representing rotations, and another one representing translations.

5.4.1 Initialization

Before defining the Dual Quaternion representation, we need to define our rigid motion composed by a 4x4 matrix in homogeneous coordinates :

$$RT = (Rx * Ry * Rz * T)$$

in which :

- Rx is a 4x4 rotation matrix corresponding to the pitch angle α :

$$Rx = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Ry is a 4x4 rotation matrix corresponding to the roll angle β :

$$Ry = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rz is a 4x4 rotation matrix corresponding to the yaw angle γ :

$$Rz = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- T is a 4x1 augmented matrix deriving from the translation vector as a 3x1 matrix corresponding x, y and z:

$$T = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

After the computation of the Rigid Motion RT we retrieved our Dual Quaternion state, defined as an 8x1 matrix.
By recalling formula to compute rotations with Quaternions [3.42]:

$$\mathbf{q} = \left[\cos\left(\frac{\hat{\theta}}{2}\right), \sin\left(\frac{\hat{\theta}}{2}\right)\hat{\mathbf{n}}\right]$$

We firstly needed to construct the Quaternion $\mathbf{q}$. Since we have Rotation matrix 3x3 (considering only first 3 rows and columns of Rigid Motion matrix), is possible, thanks to the formulation written in Quaternions Section, [3.25], to compute it.

After that it has been possible to compute the Rigid Motion by following the formulation [3.43].

$$RM = \mathbf{q} + \frac{1}{2}\varepsilon\mathbf{q}\mathbf{t}_{ON}$$

in which, the Dual Quaternion $\hat{\mathbf{q}}$, in matrix representation, corresponds to 8x1 matrix, that is the final representation of our hidden state :

$$x = \hat{\mathbf{q}} = \begin{bmatrix}\mathbf{q} \\ \mathbf{q}\mathbf{t}\end{bmatrix}$$

Regarding *Sigma points*, they are computed, by following the canonical parameters α, κ and γ.
For what concerns Covariance matrices and noise we have:

- Initial covariance Matrix P:

$$P = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.01 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.01 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.11 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.11 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.11
\end{bmatrix}$$

- Initial process noise Matrix R:

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.11 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.11 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.11 \end{bmatrix}$$

- Initial measurements noise Matrix R:

$$R = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.11 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.11 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.11 \end{bmatrix}$$

Finally, considering functions $f(\dots)$ and $g(\dots)$ that, as we have already explained, transform sigma points to the correspondent space (state space or measurement space), we concluded that can be both linear functions, since Dual Quaternions representation handle the problem of angle warping encountered in the previous experiment.

Main Operations adopted to work with Quaternions and Matrices, which have been repeatedly adopted in this experiment, were explained in section 4.

Now we take a look at plots obtained on this test.

5.4.2 Plots experiment 3

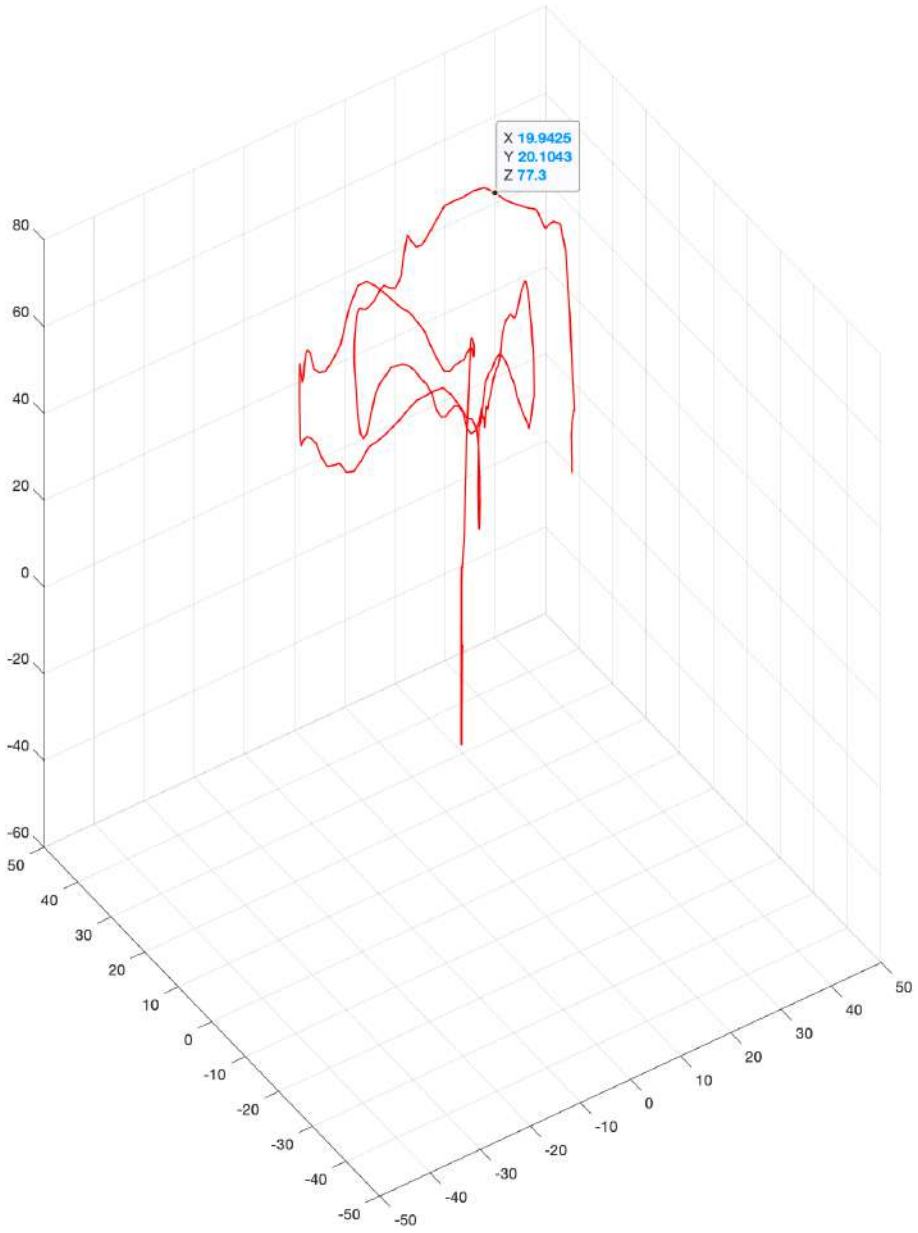

Figure 5.10: Plot of trajectory obtained by represent raw data, given only GPS position data and Altitude from Barometer sensor. No Kalman filter applied

As for the other experiments, the red line plotted on 3d graph represent the measurements acquired by the sensors. No Kalman filter was applied here.

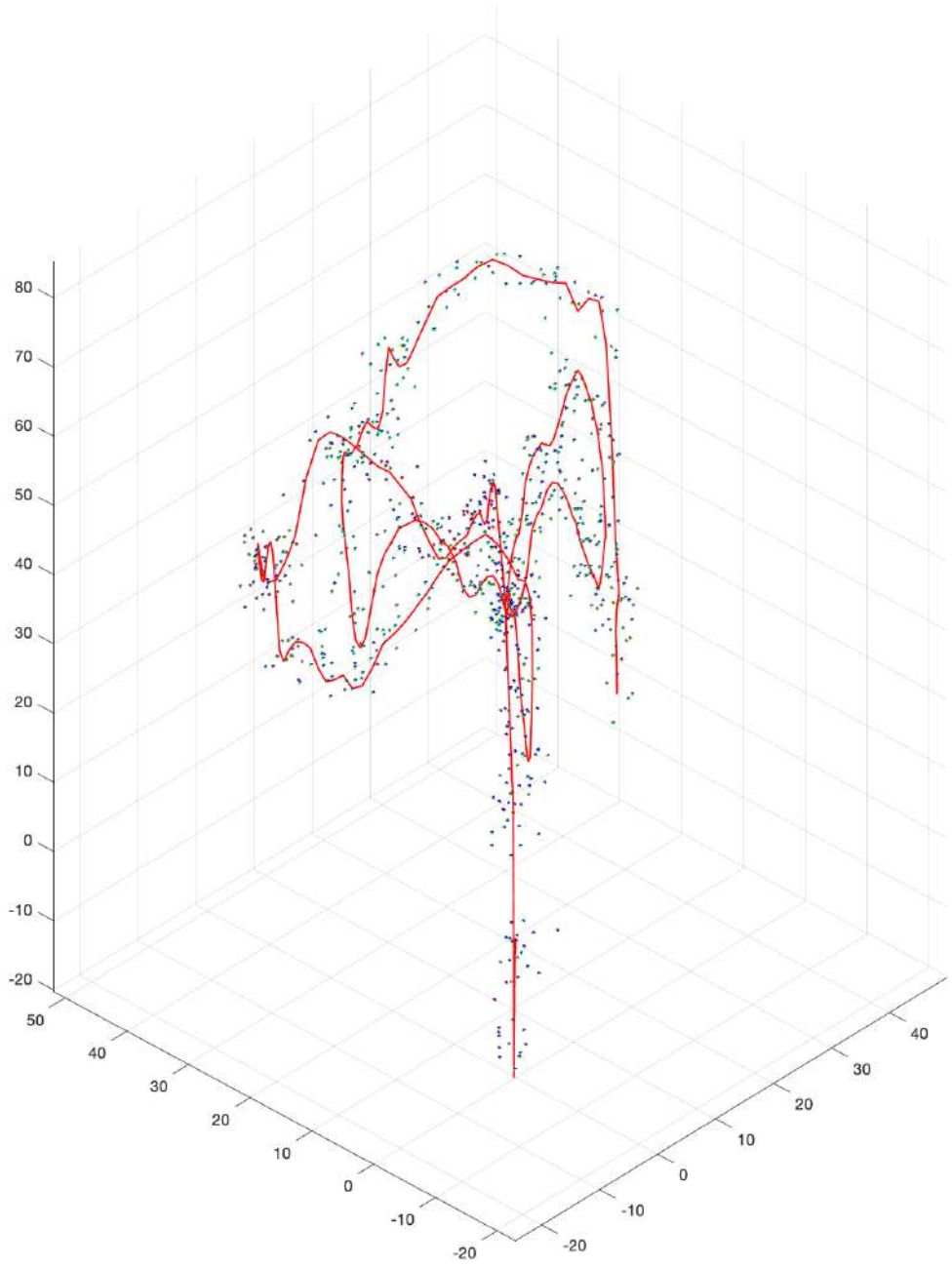

Figure 5.11: Plot of trajectory obtained by represent raw data, given only GPS position data and Altitude from Barometer sensor. No Kalman filter applied

This red line represent measurements plotted in figure [5.10] and new measurements computed by The Unscented Kalman filter in [5.11]. As we can see this measurements, represented as 3d vector having different positions and orientations, are distant from the red line, meaning that they are perturbed by some Gaussian noise. The new path is not yet plotted, and in this case all this measurement will be used to compute the new trajectory.

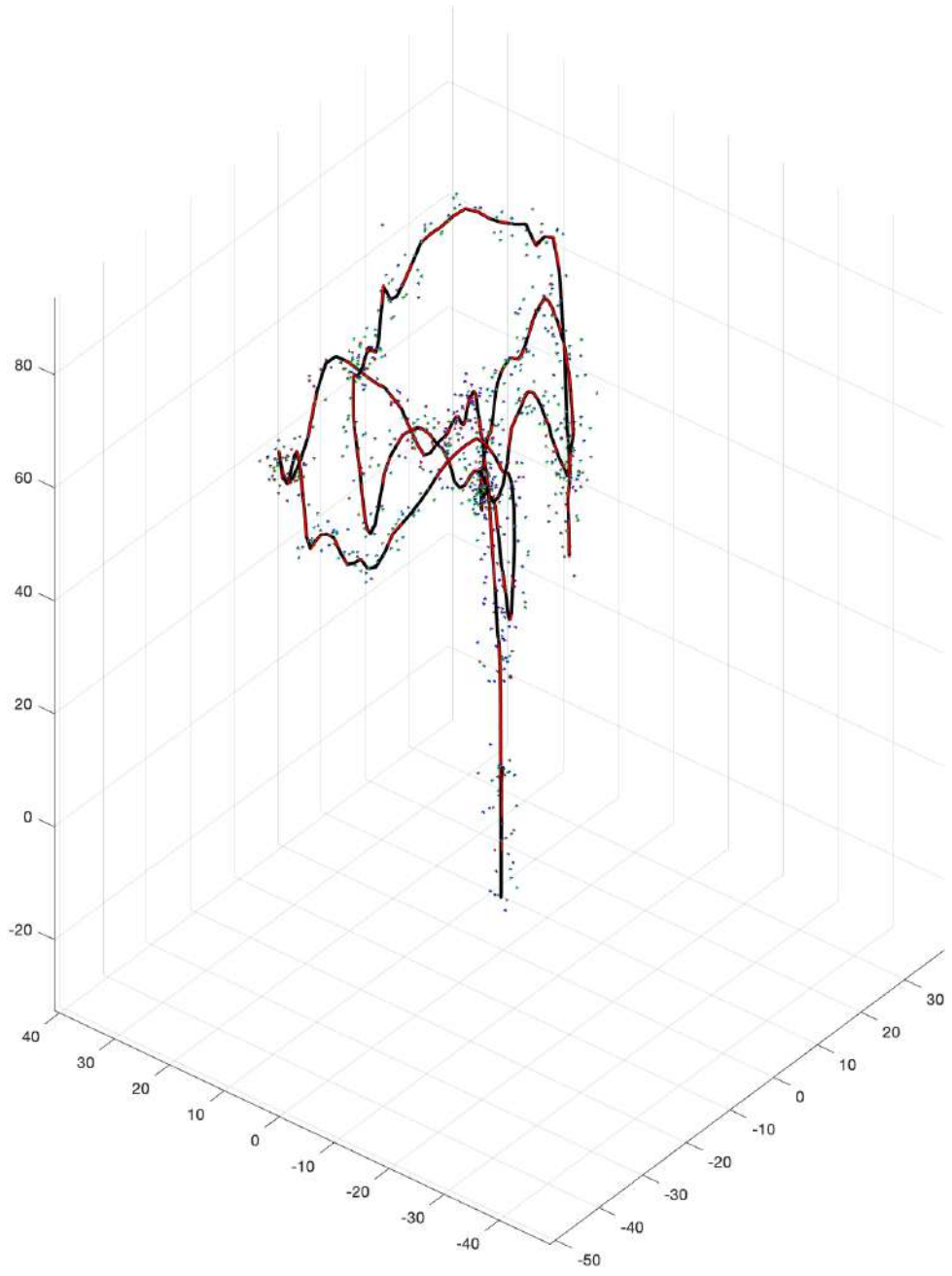

Figure 5.12: Plot of trajectory obtained by represent raw data, given only GPS position data and Altitude from Barometer sensor. No Kalman filter applied

Last plot [5.12], demonstrate how effective is the UKF with Dual Quaternions.
By considering only measurements affected by noise, it has been able to compute a very precise path similar to the red one showed in [5.10].
Last graph [5.13] in the next page shows a zoomed slice of the path drawn above. Even if the black line overlaps the red one, it possible to notice that it's smoother than the red one, meaning that the Unscented Kalman filter with Dual Quaternions perfectly fits the original model.
Once again, by modelling process noise and covariances is possible to handle the reliability of the model.

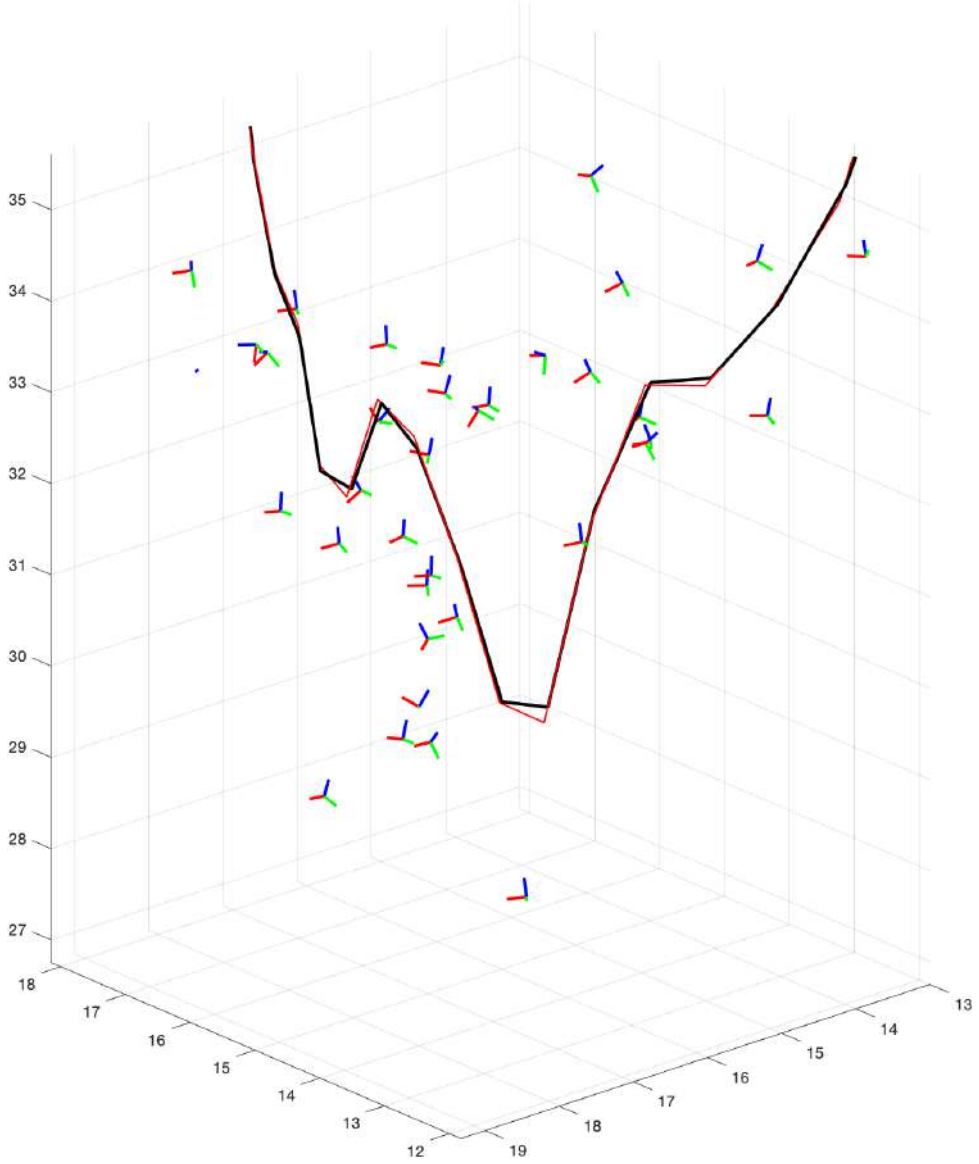

Figure 5.13: Portion of the final plot

6 Conclusions

6.1 Final considerations

This thesis has been developed with the aim to demonstrate how important how effective is the Kalman Filter.
Our main attention have been posed on the necessary backgrounds to understand the functionality of this tool. In particular sections 3 and 4 regarding the introduction of Kalman filter and Quaternion are a good point to start with.
After the introduction, we started to introduce our real experiments, by using an FPV drone and data retrieved from its sensors (IMU, COMPASS, GYRO).
On this work, the more interesting result is given by the second experiment, which shows how to handle angle warping caused by modular angles on a custom Unscented Kalman filter. In fact, thanks to the use of a non-linear function g, the so called $atan2$, we derived, from the hidden state composed by the $\sin\gamma$ and $\cos\gamma$, the angle γ in measurement space. This methodology allowed us to resolve the problem encountered in the first experiment.
In The third experiment instead, has been tested raw data in a Dual Quaternion Unscented Kalman filter. Since we were considering Unit Dual Quaternions, i.e. with norm equal to 1, belongs to a Manifold and so all the operations allowed there has been adopted, but not showed since it wasn't our main goal. This last experiment shows the utility on applying Dual Quaternions with the Unscented Kalman Filter, as a result to provide a good estimation of roto-translation computed in a single state.
Moreover, what has been explained is also shown with graphics plots that allow to understand better the behaviour and the goal of our experiments.

6.2 Future Works

On our journey on this work we thought on possible future works. Recalling that Kalman Filters are not a heavy tools in terms of computationally costs, it would have been possible to develop a real time Kalman filter, but it would have required more sophisticated acknowledge to transmit data in a wireless way, with fast transmission and without loss of information's packets.
Moreover, since in this work we have considered only position and angles rotation of our model, a future improvement could introduce new sensors, such as accelerometer, to compute the predicted position without the use of compass data, but instead measure the variance from them.
After those implementation, a final and very prestigious step could resolve on building an autopilot based on the usage of Kalman filters, by using the technology involving FPV drones, to make them more secure and reliable instruments to work with.

Thanks for your attention.

Bibliography

[1] Rafal Ablamowicz, Garret Sobczyk, et al. *Lectures on Clifford (geometric) algebras and applications*. Springer, 2004.

[2] Bruno Vilhena Adorno. "Robot Kinematic Modeling and Control Based on Dual Quaternion Algebra—Part I: Fundamentals." In: (2017).

[3] KS Amelin and AB Miller. "An algorithm for refinement of the position of a light UAV on the basis of Kalman filtering of bearing measurements". In: *Journal of Communications Technology and Electronics* 59.6 (2014), pp. 622–631.

[4] Tim Bailey et al. "Consistency of the EKF-SLAM algorithm". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2006, pp. 3562–3568.

[5] Moti Ben-Ari. "A tutorial on euler angles and quaternions". In: *Weizmann Institute of Science, Israel* 524 (2014).

[6] Steven D Brown. "The Kalman filter in analytical chemistry". In: *Analytica chimica acta* 181 (1986), pp. 1–26.

[7] Benjamin Busam, Tolga Birdal, and Nassir Navab. "Camera pose filtering with local regression geodesics on the riemannian manifold of dual quaternions". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017, pp. 2436–2445.

[8] David T Cole, Salah Sukkarieh, and Ali Haydar Göktoğan. "System development and demonstration of a UAV control architecture for information gathering missions". In: *Journal of Field Robotics* 23.6-7 (2006), pp. 417–440.

[9] Manfredo Perdigao Do Carmo and J Flaherty Francis. *Riemannian geometry*. Vol. 6. Springer, 1992.

[10] Janez Funda and Richard P Paul. "A computational analysis of screw transformations in robotics". In: *IEEE Transactions on robotics and automation* 6.3 (1990), pp. 348–356.

[11] James Samuel Goddard and Mongi A Abidi. "Pose and motion estimation using dual quaternion-based extended Kalman filtering". In: *Three-Dimensional Image Capture and Applications*. Vol. 3313. International Society for Optics and Photonics. 1998, pp. 189–200.

[12] Søren Hauberg, François Lauze, and Kim Steenstrup Pedersen. "Unscented Kalman filtering on Riemannian manifolds". In: *Journal of mathematical imaging and vision* 46.1 (2013), pp. 103–120.

[13] Kenneth W Iliff. "Parameter estimation for flight vehicles". In: *Journal of Guidance, Control, and Dynamics* 12.5 (1989), pp. 609–622.

[14] Ravindra V Jategaonkar and E Plaetschke. "Algorithms for aircraft parameter estimation accounting for process and measurement noise". In: *Journal of Aircraft* 26.4 (1989), pp. 360–372.

[15] Simon J. Julier and Jeffrey K. Uhlmann. "New extension of the Kalman filter to nonlinear systems". In: 3068 (1997). Ed. by Ivan Kadar, pp. 182–193. DOI: 10.1117/12.280797. URL: https://doi.org/10.1117/12.280797.

[16] Abhijit G Kallapur and Sreenatha G Anavatti. "UAV linear and nonlinear estimation using extended Kalman filter". In: *2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA'06)*. IEEE. 2006, pp. 250–250.

[17] R. Kálmán. "A new approach to linear filtering and prediction problems". In: *Journal of Basic Engineering* 82 (1960), pp. 35–45.

[18] *Kalman Filtering: A Simple Introduction*. https://towardsdatascience.com/kalman-filtering-a-simple-introduction-df9a84307add. Accessed: 2/01/2022.

[19] Ladislav Kavan et al. "Geometric skinning with approximate dual quaternion blending". In: *ACM Transactions on Graphics (TOG)* 27.4 (2008), pp. 1–23.

[20] Edgar Kraft. "A quaternion-based unscented Kalman filter for orientation tracking". In: *Proceedings of the sixth international conference of information fusion*. Vol. 1. 1. IEEE Cairns. 2003, pp. 47–54.

[21] Thanh Mung Lam et al. "Artificial force field for haptic feedback in UAV teleoperation". In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 39.6 (2009), pp. 1316–1330.

[22] Kailai Li, Florian Pfaff, and Uwe D Hanebeck. "Geometry-driven stochastic modeling of se (3) states based on dual quaternion representation". In: *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*. IEEE. 2019, pp. 254–260.

[23] Kailai Li, Florian Pfaff, and Uwe D Hanebeck. "Unscented Dual Quaternion Particle Filter for SE (3) Estimation". In: *IEEE Control Systems Letters* 5.2 (2020), pp. 647–652.

[24] Kailai Li et al. "Simultaneous Localization and Mapping Using a Novel Dual Quaternion Particle Filter". In: *2018 21st International Conference on Information Fusion (FUSION)*. IEEE. 2018, pp. 1668–1675.

[25] Mingyang Li and Anastasios I Mourikis. "High-precision, consistent EKF-based visual-inertial odometry". In: *The International Journal of Robotics Research* 32.6 (2013), pp. 690–711.

[26] Giuseppe Loianno, Michael Watterson, and Vijay Kumar. "Visual inertial odometry for quadrotors on SE (3)". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 1544–1551.

[27] Chunbo Luo et al. "UAV position estimation and collision avoidance using the extended Kalman filter". In: *IEEE Transactions on Vehicular Technology* 62.6 (2013), pp. 2749–2762.

[28] Guoqiang Mao, Sam Drake, and Brian DO Anderson. "Design of an extended kalman filter for uav localization". In: *2007 Information, Decision and Control*. IEEE. 2007, pp. 224–229.

[29] Cl Masreliez and R Martin. "Robust Bayesian estimation for the linear model and robustifying the Kalman filter". In: *IEEE transactions on Automatic Control* 22.3 (1977), pp. 361–371.

[30] Ivan Maza et al. "Firemen monitoring with multiple UAVs for search and rescue missions". In: *2010 IEEE Safety Security and Rescue Robotics*. IEEE. 2010, pp. 1–6.

[31] Alexander McAulay. *Octonions: A Development of Clifford's Bi-quaterions*. University Press, 1898.

[32] Leonard A McGee and Stanley F Schmidt. *Discovery of the Kalman filter as a practical tool for aerospace and industry*. Tech. rep. 1985.

[33] *Normal (Gaussian) Distribution*. `https://www.probabilitycourse.com/chapter4/4_2_3_normal.php`. Accessed: 2/01/2022.

[34] Xavier Pennec. "Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements". In: *Journal of Mathematical Imaging and Vision* 25.1 (2006), pp. 127–154.

[35] Xavier Pennec, Pierre Fillard, and Nicholas Ayache. "A Riemannian framework for tensor computing". In: *International Journal of computer vision* 66.1 (2006), pp. 41–66.

[36] E Pennestrı and R Stefanelli. "Linear algebra and numerical algorithms using dual numbers". In: *Multibody System Dynamics* 18.3 (2007), pp. 323–344.

[37] Ettore Pennestrı and Pier Paolo Valentini. "Dual quaternions as a tool for rigid body motion analysis: A tutorial with an application to biomechanics". In: *Archive of Mechanical Engineering* 57.2 (2010), pp. 187–205.

[38] Hoang-Lan Pham et al. "Position and orientation control of robot manipulators using dual quaternion feedback". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 658–663.

[39] Mark L Psiaki. "Backward-smoothing extended Kalman filter". In: *Journal of guidance, control, and dynamics* 28.5 (2005), pp. 885–894.

[40] Maria Isabel Ribeiro. "Kalman and extended kalman filters: Concept, derivation and properties". In: *Institute for Systems and Robotics* 43 (2004), p. 46.

[41] Donald T Sant. "Generalized least squares applied to time varying parameter models". In: *Annals of Economic and Social Measurement, Volume 6, number 3*. NBER, 1977, pp. 301–314.

[42] Aksel Sveier and Olav Egeland. "Dual Quaternion Particle Filtering for Pose Estimation". In: *IEEE Transactions on Control Systems Technology* 29.5 (2021), pp. 2012–2025. DOI: 10.1109/TCST.2020.3026926.

[43] WT Luke Teacy et al. "Maintaining connectivity in UAV swarm sensing". In: *2010 IEEE Globecom Workshops*. IEEE. 2010, pp. 1771–1776.

[44] Andrea Torsello, Emanuele Rodola, and Andrea Albarelli. "Multiview registration via graph diffusion of dual quaternions". In: *CVPR 2011*. IEEE. 2011, pp. 2441–2448.

[45] Rudolph Van Der Merwe et al. "The unscented particle filter". In: *Advances in neural information processing systems* 13 (2000).

[46] Eric Wan, Rudolph Van Der Merwe, and Alex Nelson. "Dual estimation and the unscented transformation". In: *Advances in neural information processing systems* 12 (1999).

[47] Eric A Wan and Rudolph Van Der Merwe. "The unscented Kalman filter for nonlinear estimation". In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. Ieee. 2000, pp. 153–158.

[48] Liang-Chuan Wang and Chih-Yu Wen. "Adaptive trajectory tracking for UAV guidance with bayesian filtering". In: *2010 5th IEEE Conference on Industrial Electronics and Applications*. IEEE. 2010, pp. 2311–2316.

[49] Yuanxin Wu et al. "Strapdown inertial navigation system algorithms based on dual quaternions". In: *Aerospace and Electronic Systems, IEEE Transactions on* 41 (Feb. 2005), pp. 110–132. DOI: 10.1109/TAES.2005.1413751.

[50] An Tzu Yang. *Application of quaternion algebra and dual numbers to the analysis of spatial mechanisms*. Columbia University, 1963.