Alma Mater Studiorum · Università di Bologna

SCUOLA DI INGEGNERIA E ARCHITETTURA

Corso di Laurea in Automation Engineering

ROTATIONS, QUATERNIONS AND POSE ESTIMATION

Relatore:
Prof.
MASSIMO FERRI

Presentata da:
FILIPPO SENZANI PEZZI

Sessione II
Anno Accademico 2018/2019

Abstract

One of the most important skills for a robot is to recognize and locate rigid bodies in 3D space. A robot needs to know how to represent a given information, what kind of paradigms can be used to process it, and how to implement those paradigms. The determination of the position and orientation of objects in the scene takes the name of pose estimation, which can be handle either in two or three dimensions. The goal of pose estimation is to answer to the following questions: How to represent models and scenes? What constraints and what paradigms are relevant?

For sure, something relevant are translations and rotations performed by a robot arm. The aim of this work, in fact, is not only to describe the pose estimation problem, which rather concerns computer vision, but also to give the right mathematical instruments to pursue this ambition. Starting from a general description of rotations in the first Chapter, with an important excursus about Euler angles, this work continues through Chapter 2 and 3 taking in consideration Lie groups, special unitary matrices and quaternions, which are all strictly bonded to rotations. Eventually, Chapter 4 analyzes the pose estimation problem and some of the solutions provided through the years.

Sommario

Una delle caratteristiche più importanti di un robot è quella di riconoscere e localizzare i corpi rigidi nello spazio tridimensionale. Un robot deve sapere come rappresentare una data informazione, che tipo di paradigmi possono essere usati per processarla e come implementare quei paradigmi. La determinazione della posizione e dell'orientamento di un oggetto nella scena prende il nome di stima della posa, che può essere praticata sia in due che in tre dimensioni. L'obiettivo della stima della posa è rispondere alle seguenti domande: come si rappresentano i modelli e le scene? Quali vincoli e quali paradigmi sono rilevanti?

Di certo, le traslazioni e le rotazioni compiute da un braccio robotico sono rilevanti. Infatti, il proposito di questa tesi non è solo quello di descrivere il problema della stima della posa, che concerne piuttosto la computer vision, ma anche fornire i giusti strumenti matematici per ottenere questo risultato.

Partendo da una descrizione generale delle rotazioni nel primo Capitolo, con un excursus importante riguardo gli angoli di Eulero, la tesi continua attraverso il Capitolo 2 e 3 prendendo in considerazione i gruppi di Lie, le matrici unitarie speciali e i quaternioni, che sono strettamente legati alle rotazioni. In conclusione, il Capitolo 4 analizza il problema della stima della posa e alcune delle soluzioni fornite nel corso degli anni.

Ringraziamenti

Prima di tutto vorrei ringraziare il Professor Massimo Ferri: la sua competenza e la sua disponibilità hanno reso leggera quest'ultima tappa del mio percorso.
Un grazie speciale a tutti i miei amici, in particolar modo a Fede, alla Betta, a Lollo e alla Claudia: i veri amici sono quelli che ti aiutano a rialzarti quando gli altri neanche sanno che sei caduto. Siete sempre stati il mio porto sicuro nei momenti di tempesta, e anche se i chilometri tra di noi in futuro potranno essere molti, vi porterò sempre nei miei pensieri.
Ma il ringraziamento più grande di tutti non può che andare a "babbomamma", ai quali dedico questo importantissimo traguardo. Mi hanno sempre sostenuto sia moralmente che economicamente in ogni mia decisione, senza mai farmi pesare niente, cercando di farmi sentire il più tranquillo e spensierato possibile. E tutto questo non è per niente scontato. "Per aspera ad astra", e se mai riuscissi davvero ad arrivare alle stelle, mi auguro di trovarvi sempre al campo base a sostenermi. Vi voglio bene.

"E quindi uscimmo a riveder le stelle."

(Inferno XXXIV, 139)

Contents

Chapter 1

Rotations and Euler angles

1.1 Introduction

Being $SO(\mathbb{R}^n)$ the special orthogonal group of $\mathbb{R}^n$, i.e. the group of unitary operators $T : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, such that $det(T) = 1$, we call rotation in the affine space $\mathbb{A}^n(\mathbb{R})$ an element of the group $SO(\mathbb{R}^n)$ [1]. Being $A \in SO(3)$, and being $A \neq Id$, we can state that A owns the eigenvalue $\lambda = 1$, and the corresponding eigenspace $\mathbb{R}^3_1 = \{v \in \mathbb{R}^3 | A(v) = v\}$ has a dimension equal to 1. Moreover, each rotation $A \in SO(3)$ leaves fixed all the points of a straight line passing through the centre of rotation O: this straight line is called rotation axis.
In general it can be stated that a rotation matrix is a linear operator which transforms the coordinates of a point from a reference frame in another one, whose origin is coincident with the starting reference frame. Rotation matrices are orthonormal matrices with a positive and unitary determinant. An orthonormal matrix is characterized by the following properties:

- the inverse and the transpose coincide: $RR' = R'R = I$, i.e. $R^{-1} = R'$;
- matrix's lines are orthogonal, i.e. perpendicular vectors with unitary norm;
- matrix's determinant has unitary module;
- rigid rotation keeps unchanged the distances between each couple of points and angles among segments, that is, the dot product is invariant with respect to rotation: $(p_1 R) \cdot (p_2 R) = p_1 \cdot p_2$

In a 3D space, rotations form a non commutative group with respect to the product between matrices. This group is called special group of orthonormal rotation and it's indicated with $SO(3)$. Actually, the representation by means of rotation matrices may present some inconvenients:

first of all it's not so intuitive to find out which are the three independent parameters which characterize a generic matrix. Second, it's not numerically robust, which means that the procedure carried out to calculate the values of a matrix may lead to a result which doesn't belong to $SO(3)$, so that matrix needs to be orthogonalized.

1.2 Rotations

Even though they are not a minimal representation as Euler angles, rotation matrices are a valuable instrument to represent the orientation of a rigid body in space [2]. As a consequence, it would be of interest to convert an orientation expressed by means of Euler angles in the equivalent rotation matrix, and viceversa.

Rotation is a particular kind of isometric transformation which moves points of a rigid body in space, without changing distance between themselves, around the rotation axis, whose points don't change position. Consider, as an example, a rotation around z-axis: each point of the rotating body moves on a plane parallel to x-axis and y-axis. Given P the initial point and Q the point obtained after the rotation around z of an angle θ, the components of the two points would be $X_P = \rho\cos\phi$ and $Y_P = \rho\sin\phi$, $X_Q = \rho\cos(\phi+\theta)$ and $Y_Q = \rho\sin(\phi+\theta)$.

Operating in homogeneus coordinates, it's possible to find a matrix H such that $Q = HP$. After some calculation, we can state that

$$H = Rot(z,\theta) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly, for rotation around y-axis and x-axis,

$$Rot(y,\theta) = \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot(x,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1.2.1 Angle and axis of an orthogonal matrix

We shall be concerned by the parametrization of a rotation in the form $R(\phi\mathbf{n})$, where ϕ is in a range from $-\pi$ to π, and $\mathbf{n}$ is a unitary vector. The problem we want to solve is, given the matrix $A \in SO(3)$, to determine ϕ,$\mathbf{n}$.

Since $\cos\phi = \frac{1}{2}(Tr\ A - 1)$, we must find the solution of the characteristic equation for A corresponding to the root $+1$, which is the eigenvalue that corresponds to the eigenvector $\mathbf{n}$.

Every matrix can be written as the sum of a symmetric and a skew-symmetric matrix:

$A = \frac{1}{2}\{(A + A') + (A - A')\}$. It's convenient to call S the skew-symmetric "component" $A - A'$:

$$S = A - A' = \begin{bmatrix} 0 & c & b \\ -c & 0 & a \\ -b & -a & 0 \end{bmatrix}$$

Thanks to the orthogonality condition $AA' = 1$, the equation to be solved is $S\mathbf{n} = 0$, which could be written as

$$\begin{bmatrix} 0 & c & b \\ -c & 0 & a \\ -b & -a & 0 \end{bmatrix} \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = 0$$

Now, we can assume that $n_x = an_z/c$ and $n_y = an_z/c$, where $n_z = \pm c(a^2 + b^2 + c^2)^{-\frac{1}{2}}$. Considering that

$$a^2 + b^2 + c^2 = \frac{1}{2}Tr(SS') = 4\sin^2\phi$$

we can find $n_x = \pm a(2sin\phi)^{-1}$, $n_y = \mp b(2sin\phi)^{-1}$, $n_z = \pm c(2sin\phi)^{-1}$.

1.2.2 The matrix of rotation $R(\phi\mathbf{n})$

Given the rotation angle ϕ and the rotation axis $\mathbf{n} = (n_x n_y n_z)$, the corresponding orthogonal matrix is requested. As we already know, given any skew-symmetric matrix S, an orthogonal matrix A can be built up by means of the formula $A = expS = 1 + S + \frac{1}{2!}S^2 + ...$, and then $A\mathbf{n}$ $= \mathbf{n} + S\mathbf{n} + \frac{1}{2!}S^2\mathbf{n} + ... = \mathbf{n}$, because $S^m\mathbf{n} = 0$, $\forall m$.

By redefining the skew-symmetric matrix as

$$Z = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix}$$

we can continue to state that

$$\mathbf{A} = exp(\phi Z) = \mathbf{1} + (\sin\phi)Z + (1 - \cos\phi)Z^2$$

Knowing that $Tr\ Z = 0$ and $Tr\ Z^2 = -2$, $Tr\ \mathbf{A} = 1 + 2\cos\phi$, which shows that $\mathbf{A}$ coincides with the desired matrix A:

$$A = \hat{R}_r^1(\phi\mathbf{n}) = \mathbf{1} + \sin\phi Z + 2(\sin^2\frac{1}{2}\phi)Z^2$$

Thus, the matrix product that we shall need is easily obtained: $Z\mathbf{r} = \mathbf{n} \times \mathbf{r}$. In conclusion, we find

$$R(\phi\mathbf{n})\mathbf{r} = \mathbf{r} + (\sin\phi)(\mathbf{n} \times \mathbf{r}) + 2(\sin^2\frac{1}{2}\phi)\mathbf{n} \times (\mathbf{n} \times \mathbf{r})$$

This is known as conical transformation of $\mathbf{r}$ because under it this vector moves on a cone around the rotation axis $\mathbf{n}$.

1.3 Euler angles

In this section, the problem of representing univocally the position of a rigid body in space is considered. By means of a reference frame in homogeneus coordinates, it's possible to fix some appropriate conventions (e.g. axes origin in space) and therefore determine the position and the orientation of the rigid body. Consider, for example, a rigid body and two reference frames, one fixed in space and one fixed to the body (that means, a reference frame whose origin is coincident with the center of mass of the body). It can be proved that any position in space of that body can be described by means of a rotation and of a translation. Why did I consider two reference frames? Because both rotation and translation are expressed as transformations of the base frame of the body with respect to the fixed one [3].
Each rototranslation of a rigid body in space may be represented by the following relation:

$$q' = Rq + d$$

where q and q' represent the body (or, better, each point of the body) respectively before and after the rototranslation; d represents the translation and R the rotation. It's worth mentioning that a rigid motion is called rototranslation if there exists a direction (called prime direction) of fixed straight lines which, during the motion, remains parallel to itself. The direction of angular velocity is the same of the prime direction, and a fixed straight line is a straight line which passes through two points of the rigid body (thus, there exists an infinite number of fixed straight lines). Whilst vector d contains 3 parameters, rotation matrix R contains 9 parameters. That's because a rigid body in space has 6 degrees of freedom (DOFs): 3 DOFs for rotation and 3 DOFs for translation. Then, vector d is a minimal representation of the translation of the body, because it contains a number of independent variables equal to the number of DOFs

(i.e. 3). On the other hand, R is not a minimal representation of the rotation, because it utilizes 9 parameters in order to represent 3 DOFs. Therefore those 9 parameters are linearly dependent and they could be brought back to the minimal 3. Indeed, it can be proved that the 9 parameters of R are bonded to the relation

$$RR' = I$$

which, alone, fixes 6 parameters. The 3 left of course are those fixed by the effective degrees of rotation. If the goal is to proceed with the use of a minimal representation, I need to change the way I represent a rigid body in space. More precisely, I need to replace R with something minimal, something which uses only 3 parameters: Euler angles answers to this requirement.

1.3.1 Euler angles: a minimal representation of orientation

The representation of orientation in space is a complex issue. Euler's rotation theorem states that, in 3D space, any displacement of a rigid body in such way that a point on the rigid body remains fixed is equivalent to a single rotation about an axis that passes through the fixed point. Accordingly, such rotation can be described by three independent parameters: two for describing the axis and one for the rotation angle. Orientation in space, however, can be represented in several other ways, each with its own advantages and disadvantages. Some of these representations use more than the necessary minimum of three parameters.
As in the case of rotation matrix R, also in case of Euler angles it's necessary to use a reference frame integral to the rigid body; but, unlike R, the final rotation is described in a minimal way through 3 rotations around coordinate axes. The only limitation is that two consecutive rotations must take place on different axes, not around the same one. As a consequence, different combinations may be achieved, and indeed each one of them is a particular case of Euler angle. Therefore, before using them, it's necessary to fix a convention, i.e. a triad of axes with respect to the bond considered. But this triad is not the only element to be decided a priori.
Rotations through the three chosen axes may happen in two different ways. If they are all referred to the axes of the fixed frame, we speak of extrinsic rotation. If instead rotation are referred to the frame integral to the body, we are in presence of an intrinsic rotation. These two methods are ideed equivalent, i.e. an extrinsic triad may be easily lead to an intrinsic one by inverting the order of elementary rotations.

1.3.2 Triads

There exists a convention which states that Euler triads (a set of three elements, in this case three axes around which the body rotates in sequence) are only those in which the first axis is coincident with the last one. All those triads which consist of any combination of the three distinct axes are called Tait-Bryan triads (they will be considered later in this chapter).

$$\text{Euler triads } \{z-x-z; x-y-x; y-z-y; z-y-z; x-z-x; y-x-y\}$$

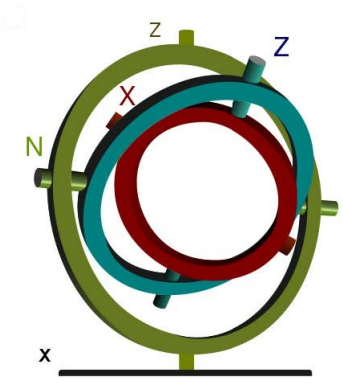
The order in which the three rotations are done is important; thus, we have a total of 216 (6^3) possible sequences. It's good to recall that each triad may be indifferently referred to intrinsic or extrinsic rotations, hence there will be 24 triads (12 exrinsic and 12 intrinsic).

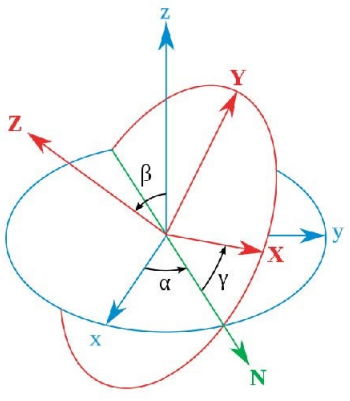
1.3.3 Euler triad Z-X-Z

Given α , β and γ angles in radians, the three rotation to be executed are the following:

1. first rotation of α around z-axis of the frame integral to the body;
2. second rotation of β around the new x-axis of the rotated reference frame (x');
3. third rotation of γ around the new z-axis of the rotated reference frame (z'').

It's possible to visualize these three rotations by means of a gyroscope:





Nodes line, Precession, Nutation and Spin

Consider XYZ the reference frame fixed to the body, and xyz the frame fixed in space: the nodes line N will be the line which intersects the two planes XY and xy, when they are distinct. If they coincide, N will coincide with X-axis. Thanks to this definition, now it's possible to provide the geometric definition of Euler angles:

- α, the angle between x-axis and N, is called angle of *precession* ($0 \leq \alpha < 2\pi$);
- β, the angle between z-axis and Z-axis, il called angle of *nutation* ($0 < \beta < \pi$);
- γ, the angle between N and X-axis, is called angle of *spin* ($0 \leq \gamma < 2\pi$).

Usually, α, β and γ are replaced respectively by ψ, θ and ϕ, angles in radians as well.
Nodes line could be represented as the ratio

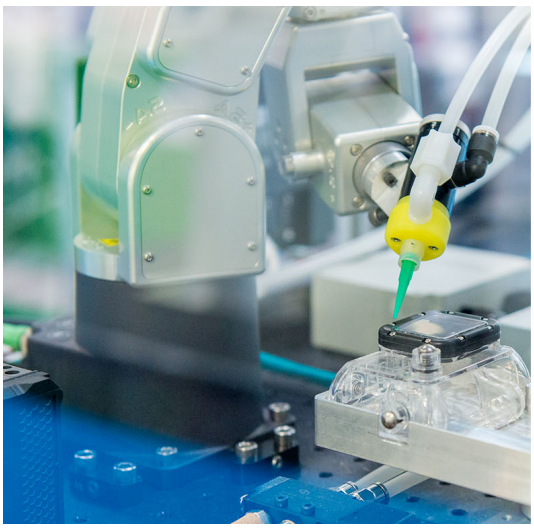
$$\vec{n}(t) = \frac{\vec{e}_3(0) \times \vec{e}_3(t)}{|\vec{e}_3(0) \times \vec{e}_3(t)|}$$

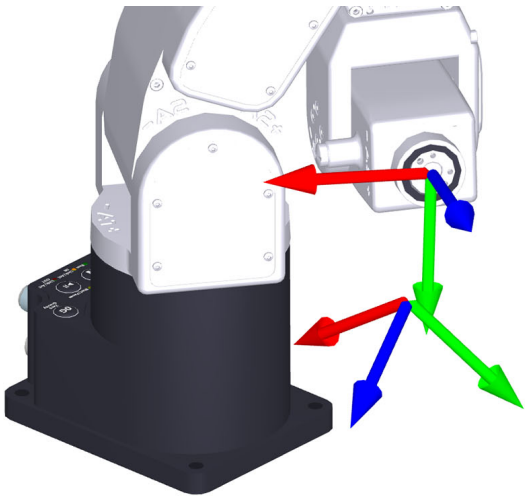
where $\vec{e}_3(0)$ is the versor along z-axis of a fixed frame, while $\vec{e}_3(t)$ is the versor along z-axis of the frame fixed to the body.
The angular velocity of a rigid body may be represented as a function of Euler angles:

$$\vec{\omega} = \vec{\omega}(\dot{\psi}, \dot{\theta}, \dot{\phi}, \psi, \theta, \phi)$$

1.3.4 Example of application of Euler angles

Consider the following real-life situation: it's requested to attach a *FISNAR* dispensing valve to the end-effector of a *Meca500* robot arm. Naturally, the engineer who designed and machined the adapter didn't care about Euler angles and was only concerned with machinability and reachability. In his design, there were essentially two rotations of 45°. Firstly, he used two diametrically opposite threaded holes on the robot flange to attach the adapter, which caused the first rotation of 45°. Secondly, the angle between the flange interface plane and the axis of the dispenser was 45°.

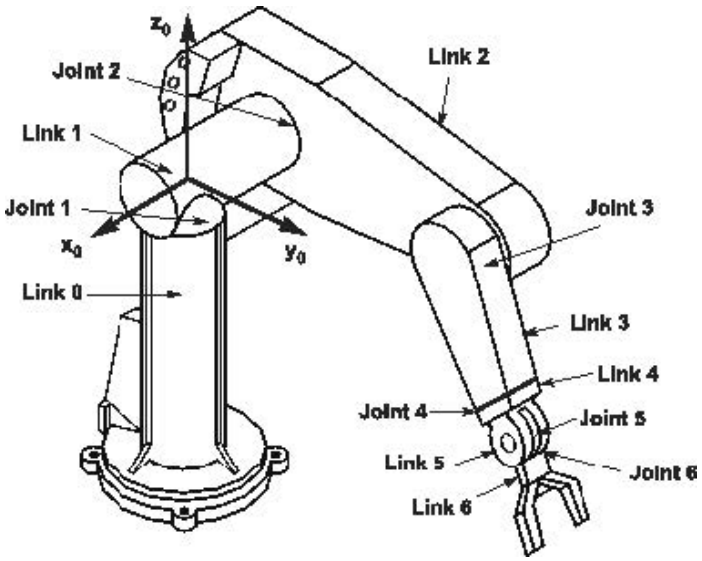




Note that when using axi-symmetric tools, it is a common practice to align the tool z-axis with the axis of the tool. This is particularly useful with the mobile xyz Euler angle convention, since the redundant rotation about the axi-symmetric tool corresponds to the third Euler angle, γ. Thus, the first two Euler angles define the axis of the tool, while the third one can be used to choose the optimal configuration of the robot (i.e., far from singularities).

1.4 Euler angles and robotics

Robots kinematics studies the bond among independent variables of robot's joints and the position of joints themselves. An industrial robot may be schematized as a sequence of links connected by joints. The purpose of joints is to provide necessary DOFs in order to reach specific points in surrounding space.



The last link (Link 6 in figure) is the end effector, in charge of the manipulation (grab and release) of objects. The first question to be asked is how many joint/link couples are essential for catching any object in space. Nowadays, robots are built so that each point of space could be reached by the end effector by means of any orientation: six DOFs will be necessary, therefore the end effector could be considered as a rigid body. It's possible then to say that the number of joint/link couples necessary to move the end effector toward any point is 6.
From now on, we shall only consider end effectors with 6 DOFs inside robot manipulators with 6 pairs of joints/links.

1.4.1 A more natural representation by means of Euler angles

Consider to separate the 3 DOFs of translation from the 3 DOFs of rotation. This representation brings remarkable semplifications, from the point of view of both modeling and control. Thanks to a more rigorous representation, the position of the effector in space could be described by means of vector v:

$$v = \begin{bmatrix} p \\ \phi \end{bmatrix}$$

where $p \in R_3$ represents the position of the effector, while ϕ represents its orientation.
Vector p consists of a triple of real values, so it can be thought of as a point in space indicated by x, y, z. For what concerns ϕ, it should be a minimal representation; in fact a rotation matrix may bring some inconvenience. The minimal representation of orientation which perfectly fits this role is given by Euler (or Tait-Bryan) angles.
In order to use an Euler triad, a fixed and a mobile frame should be chosen; the mobile one has to be fixed to the object which rotates. As fas as the fixed frame is concerned, usually it's fixed at the base of the robot.
At this point, it has to be decided which Euler or Tait-Bryan triad should be used to represent the rotations; it's convenient to choose the one which best reflects movements of robot's wirst. In a robot manipulator the last three joints has the task to provide the final rotation of the effector by means of three independent and orthogonal rotations.

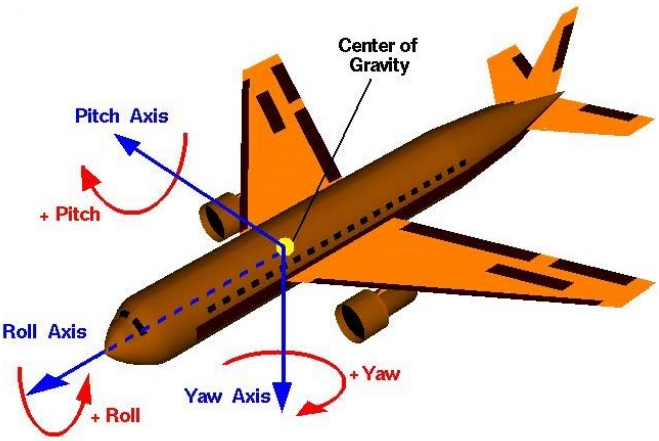
1.4.2 Pros & Cons of Euler angles

Euler angles solve the problem of the mapping of DOFs, indeed they express the desired rotation as a sequence of three different rotations around default axes, so we have 3 angles in 3 DOFs.

Moreover, they are numerically stable. Nevertheless, the geometry of orientations is difficult, and it varies with the selection of initial axes. There is no rational method to multiplicate or to combine two rotations; also the conversion between rotation matrices and angular coordinates is complicated and expensive. Besides, it's not possible to simply rotate, except with respect to default axes. In conclusion, by using Euler angles, the problem of gimbal lock may occur, which consints in the loss of a rotational DOF when two of the three axes are guided in aligned configurations.
Quaternions (see Ch. 3) aren't affected by gimbal lock problem; at the same time they preserve a compact parametrization of rotations. With respect to matrices, quaternions allow to create fluid interpolations, for example between two default angular values.

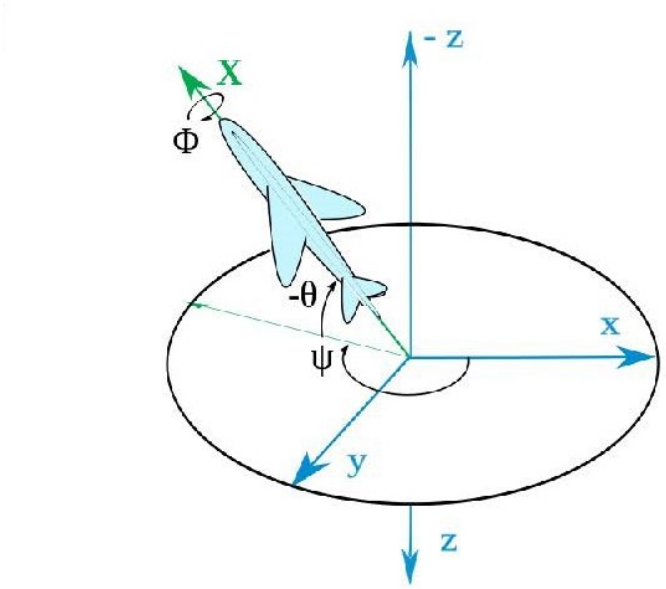
1.5 Tait-Bryan angles

Among Tait-Bryan triads $(x-y-z, y-z-x, z-x-y, x-z-y, z-y-x, y-x-z)$, the most interesting is $z-y-x$, also known as Roll-Pitch-Yaw triad, widely used in aeronautic field. As for all the other kind of triads of axes, it's necessary to choose whether intrinsic or extrinsic rotations should be carried on. By considering intrinsic rotations, we'll obtain a reference frame fixed to the rigid body: in case it's an aeroplane, the centre of the frame would coincide with the centre of gravity of the airplane, as shown in figure:



Rotation around the three axes allows the airplane to compute respectively movements of roll around z-axis, pitch around y-axis and yaw around x-axis.
Consider now to define rotations by means of an extrinsic method, relatively to a fixed reference frame $z-x-y$, which could be considered, for example, integral to Earth. The figure shows

this last case.



Also with extrinsic rotations exists a reference frame integral to the airplane, even though the three angles are all represented with respect to the fixed frame.
Going on by choosing as rotation modality the extrinsic one, given α, β, γ as the three Tait-Bryan angles, the following three rotations should be carried out to bring the body to the correct orientation:

1. rotation of α around x-axis of the fixed frame;
2. rotation of β around y-axis of the fixed frame;
3. rotation of γ around z-axis of the fixed frame.

Unlike the preceding case (precession-nutation-spin with intrinsic rotation) the order $x - y - z$ is opposite with respect to $z - x - y$, because of extrinsic rotations.

1.6 Rotation matrices and Euler triads

1.6.1 From Euler triad to rotation matrix

On one side, Euler angles describes the orientation as a consequence of three rotations around the three orthogonal axes; on the other side, we are able to represent the rotation matrix relative to an elementary rotation around one axis. Then, in order to obtain the final orientation

considering all the three rotations, it's sufficient to calculate first the three rotation matrices, then combine (multiply) them. In particular, the three rotation matrices must be multiplied following the same order of their corresponding rotation axes which appear inside the triad.
Consider now, as an example, to convert the triad $z - y - z$ of intrinsic rotations (precession - nutation - spin). The task consists in finding the three rotation matrices $Z(\alpha)$, $Y(\beta)$ and $Z(\gamma)$: it would be sufficient to apply general formulas of rotation matrices around z-axis and y-axis reported in Section 2.1, and substituting α, β and γ instead of ϕ [5]. The final rotation matrix would be

$$R(\alpha, \beta, \gamma) = Z(\alpha)Y(\beta)Z(\gamma) = Rot(Z, \alpha)Rot(Y, \beta)Rot(Z, \gamma)$$

1.6.2 From rotation matrix to Euler triad

The inverse problem is far more complicated than the preceding one. In fact, unlike the universal method we provided in the previous case, it doesn't exist a unique and precised sequence of steps in order to achieve the conversion from matrix to triad. Usually, the followed path consists in analyzing how the rotation matrix is presented, where each element is made up of products of sines and cosines, trying to find some relations which can lead to the value of angles.
Consider again the example with the triad $z - y - z$: first of all, we need to introduce the function $atan2(y, x)$, which associates to y and x the relative value of arctan $\frac{y}{x}$. The advntage of $atan2$ is that it's able to determine univocally the angle by analyzing the sign of x and y, without an uncertainity between I and III quadrant or between II and IV. Consider

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}$$

And now consider the following formulas in order to find α, β and γ:

$$\beta = atan2\left(\sqrt{R_{31}^2 + R_{32}^2}, R_{33}\right)$$

if $\sin\beta \neq 0$

$$\gamma = atan2\left(R_{32}, -R_{31}\right)$$

$$\alpha = atan2\left(R_{23}, R_{13}\right)$$

if $\sin\beta = 0$

$$\gamma = 0$$

$$\alpha = atan2\left(R_{21}, R_{11}\right)$$

In order to understand the particular case $\sin\beta = 0$, and then $\beta = 0$, we need to consider the specific triad $z - y - z$: if β is null, the rotation corresponding to y-axis is null, and then also the third rotation around z-axis would take place around the first z-axis. Thus, there exists infinitely many pairs (α, γ) which are able to provide the same final rotation. We proceed by fixing arbitrarly one of the two angles, in this case γ at 0 for convenience, and α would be found easily.
As a final consideration, it's worth pointing out again that these calculations are valid only for the $z - y - z$ triad, so the triad has to be always specified before starting the conversion. However, given a rotation matrix R, it's always possible to decide which triad one wishes to pass through.

1.6.3 From one Euler triad to another Euler triad

If more set of formulas like the one previously written for the conversion from matrix to triad are available, each one to convert R in a different triad, it's possible to use R as common point in order to convert two triads among them. If, for example, the Tait-Bryan triad $x-y-z$ needs to be converted in the Euler triad $z - x - z$, it's sufficient to follow these two steps:

1. convert $x - y - z$ into R;
2. convert R into $z - x - z$.

Chapter 2

Lie groups and SU(2) groups

2.1 Lie groups

A Lie group is a group G that is also a differentiable manifold, compatible with the operations which characterize groups [7]. That is, the inversion map

$$s : G \to G$$

$$g \mapsto g^{-1}$$

and multiplication map

$$m : G \times G \to G$$

$$(g, h) \mapsto gh$$

are smooth.

The word "smooth" in the definition above can be understood in different ways: C^1, C^∞, analytic. It turns out that all of them are equivalent: every C^0 Lie group has a unique analytic structure. This is a highly non-trivial result (it was one of Hilbert's 20 problems), and it will not be proved in this treatment.

Roughly speaking, a Lie group is a continuous group, which means that all its elements can be described by several real parameters; a group of transformations depends on these parameters in a continuous way.

Suppose G is a Lie group and M is a C^∞ manifold. An action of G on M is a map
$A : G \times M \to M$ such that

$$A(g(A(h, m))) = A(gh, m)$$

A more common notation for the action is $g \cdot m$; in these terms, this condition is
$g \cdot (h \cdot m) = (gh) \cdot m$.

Suppose G, H are Lie groups. A homomorphism from G to H is defined as a C^∞ map $F : G \to H$ which is also a group homomorphism.
Suppose G is a Lie group. A subgroup H of G in the algebraic sense is defined as a Lie subgroup if $H \subset G$ is a locally (with respect to H) closed submanifold of G.
Then, we can state that a Lie subgroup is a Lie group when equipped with the induced C^∞ structure and the inclusion $H \to G$ is a Lie group homomorphism.
Let G be a Lie group of dimension n and $H \subset G$ a Lie subgroup of dimension k. Then the coset space G/H has a natural structure of a manifold of dimension $n-k$ such that the canonical map $p : G \to G/H$ is a fiber bundle, with fiber diffeomorphic to H. The tangent space at $\bar{1} = p(1)$ is given by $T_{\bar{1}}(G/H) = T_1G/T_1H$. Let x be a point in an n-dimensional compact manifold M, and attach at x a copy of $\mathbb{R}^n$ tangential to M. The resulting structure is called the tangent space of M at x and is denoted T_xM. If γ is a smooth curve passing through x, then the derivative of γ at x is a vector in T_xM.
If H is a normal Lie subgroup then G/H has a canonical structure of a Lie group.
The following are examples of Lie groups [10] :

- $\mathbb{R}^n$, $+$;
- $\mathbb{R}^*$, $\times$;
- $\mathbb{R}_+$, $\times$;
- $S^1 = \{z \in \mathbb{C} : |z| = 1\}$, $\times$;
- $\mathrm{GL}(n, \mathbb{R}) \subset \mathbb{R}^{n^2}$;
- SU(2)=$\{A \in \mathrm{GL}(2, \mathbb{C}) | A\bar{A}^t = 1, detA = 1\}$

2.1.1 Lie algebra of a Lie group

Let G be a Lie group, and let g be a generic element of G. We define with $L_g : G \to G$ the left multiplication $L_g(h) := g \cdot h$; this is an isomorphism, thus there exists an inverse $L_g^{-1} = L_{g^{-1}}$.
At the same time, it's possible to define a right multiplication $R_g(h) := h \cdot g$ with an inverse $D_g^{-1} = D_{g^{-1}}$.
Consider now fields $\mathbf{v}$ of vectors on G; $\mathbf{v}$ could be left invariant or right invariant if, $\forall g, h \in G$, $(dL_g)(\mathbf{v}_h)] = \mathbf{v}_{S_g(h)} = \mathbf{v}_{g \cdot h}$ and $(dR_g)(\mathbf{v}_h)] = \mathbf{v}_{R_g(h)} = \mathbf{v}_{h \cdot g}$.
The set of left invariant fields is isomorphic to the set of right invariant fields for the same Lie group, even though the two sets, in general, don't coincide. The set of left (right) invariant

fields is a vectorial space for a determinated Lie group.
Suppose G is a closed Lie subgroup of $\mathrm{GL}(n,\mathbb{C})$; then the Lie algebra of G may be computed as $\mathrm{Lie}(G) = \{X \in M(n,\mathbb{C}) | \exp(tX) \in G \text{ for all } t \text{ in } \mathbb{R}\}$.

2.2 Special Unitary matrices. The SU(2) group

If a matrix has to deal with a proper rotation, it has to be orthogonal or unitary (according to whether the space on which they act is real or complex) and it has to have a unit determinant. Matrices which satisfy this last condition are known as unimodular. We shall consider now unimodular and unitary matrices of bilinear transformations. Unimodular matrices are also called "special",whence the matrices in questions are special unitary matrices of dimension two, in short SU(2) [2].
A very strong condition on the matrix elements of a SU(2) matrix is imposed by the unitary condition, which requires $A^{\dagger} = A^{-1}$. Every unitary matrix A can be normalized so that it becomes unimodular ($\det A$=1) by multiplying it with a constant, which doesn't affect the bilinear transformation:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \longmapsto \begin{bmatrix} \lambda a & \lambda b \\ \lambda c & \lambda d \end{bmatrix}$$

where $\lambda = \pm(\det A)^{-1/2}$. Moreover, every SU(2) matrix can still be multiplied by ± 1; this doesn't affect the bilinear transformation entailed by it or the unimodular condition. Therefore, A and $-A$ are interchangeable. Due to this fact, SU(2) matrix has to be preceded by the $\pm$ sign.

2.2.1 Rotations and SU(2)

In order to show how the bilinear transformation with SU(2) matrices gives 3D rotations, the concept of isotropic vector has to be introduced.
Given two unit orthogonal vectors $\mathbf{v}$ and $\mathbf{w}$, $\mathbf{v}^2 = \mathbf{w}^2 = 1$ and $\mathbf{v} \cdot \mathbf{w} = 0$. An orthogonal transformation keeps this couple of relations invariant: it can be formulated also in a more compact way by inverting a vector $\mathbf{r} = \mathbf{v} + i\mathbf{w}$, for which $\mathbf{r}^2 = \mathbf{v}^2 - \mathbf{w}^2 + 2i\mathbf{v} \cdot \mathbf{w} = 0$. Such vectors of zero length are called isotropic vectors; their main use is that they embody the couple of conditions reported before in a single equation, so that an orthogonal transformation must always transform an isotropic vector into another isotropic vector.
If we write an isotropic vector through its components x, y, and z, it would be $x^2 + y^2 + z^2 = 0$.

It's very convenient to define an isotropic parameter $w = (x - iy)z^{-1}$, for which $w^* = -w^{-1}$. The condition for SU(2) matrix to be a rotation is that $ww^* = -1$ is invariant under the rotation. Thus, given

$$\bar{w}\bar{w}^* = \frac{aw + b}{-b^*w + a^*} \cdot \frac{a^*w^* + b^*}{-bw^* + a} = \frac{aa^*ww^* + ab^*w + a^*bw^* + bb^*}{bb^*ww^* - ab^*w - a^*bw^* + aa^*}$$

where

$$\bar{w} = \begin{bmatrix} a & b \\ -b^* & a^* \end{bmatrix} \circ w, \qquad \bar{w}^* = \begin{bmatrix} a^* & b^* \\ -b & a \end{bmatrix} \circ w^*$$

we have the condition

$$\frac{-aa^* + ab^*w + a^*bw^* + bb^*}{-bb^* - ab^*w - a^*bw^* + aa^*} = -1$$

which can be satisfied only if $aa^* = 1$ and $b = 0$ (Case 1), or if $a = 0$ and $bb^* = 1$ (Case 2). These conditions are particularly simple to apply to three types of rotations: (i) all rotations $R(\phi\mathbf{z})$ around **z**-axis; (ii) a binary rotation $R(\pi\mathbf{x})$ around **x**-axis; (iii) a binary rotation $R(\pi\mathbf{y})$ around **y**-axis.

Now we see how these rotations actually emerge from the SU(2) transformations when cases 1 and 2 above are considered.

Case 1:

$$\bar{w} = aw(a^*)^{-1} = a^2w$$

This is the eigenvalue equation for the variable u_1 under a rotation $R(\phi\mathbf{z})$. In fact, $\bar{u}_1 = R(\phi\mathbf{z})u_1 = \exp(-i\phi)u_1$. The corresponding SU(2) matrix would be

$$\hat{R}(\phi\mathbf{z}) = \begin{bmatrix} e^{-i\frac{1}{2}\phi} & \\ & e^{i\frac{1}{2}\phi} \end{bmatrix}$$

Case 2:

$$\bar{w} = b(-b^*w)^{-1} = b\mathbf{2}w^{\mathbf{*}}$$

Compare this result with the following transformations

$$\bar{w} = R(\pi\mathbf{x})\{(x - iy)z^{-1}\} = (x + iy)(-z)^{-1} = -w^*$$

$$\bar{w} = R(\pi\mathbf{y})\{(x - iy)z^{-1}\} = (-x - iy)(-z)^{-1} = w^*$$

Thus these two matrices are obtained:

$$\hat{R}(\pi\mathbf{x}) = \begin{bmatrix} & -i \\ -i & \end{bmatrix}, \qquad \hat{R}(\pi\mathbf{y}) = \begin{bmatrix} & -1 \\ -1 & \end{bmatrix}$$

2.2.2 SO(3), SU(2) and quaternions

Given Q the group of quaternions (see Ch. 3) and $q = (s, \mathbf{v})$ a normalized quaternion (with $s + v^2 = 1$), SO(3) maps onto Q as follows:

$$g \in SO(3) \qquad g \mapsto \pm q, \qquad q \in Q$$

This precisely parallels the homomorphism between SU(2) and SO(3)

$$g \in SO(3) \qquad g \mapsto \pm v, \qquad v \in SU(2)$$

In fact, SU(2) and Q are isomorphic. In order to establish more precisely this isomorphism, consider the additive expression for a quaternion q: $q = (s, \mathbf{v}) = s1 + v_x i + v_y j + v_z k$. It follows that the quaternions 1, i, j, k multiply precisely as the SU(2) matrices for the identity $\mathbf{I}$, and the three bilateral binary rotations of $\mathbf{D}_2$, $\mathbf{I}_x$, $\mathbf{I}_y$, $\mathbf{I}_z$, where

$$\mathbf{I} = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix} \qquad \mathbf{I}_x = \begin{bmatrix} & -i \\ -i & \end{bmatrix} \qquad \mathbf{I}_y = \begin{bmatrix} & -1 \\ 1 & \end{bmatrix} \qquad \mathbf{I}_z = \begin{bmatrix} -i & \\ & i \end{bmatrix}$$

The group $\mathbf{D}_2$ consists of the identity E and the three bilateral binary (BB) rotations C_{2x}, C_{2y}, C_{2z}. A BB rotation is a rotation $R(\pi\mathbf{n})$ such that there exists in the group to which it belongs another operation $R(\pi\mathbf{n}')$ with $\mathbf{n}' \perp \mathbf{n}$.

Therefore, the quaternion q may be represented by a matrix $\check{A}$ obtained by replacing 1, i, j, k in that equation by $\mathbf{I}$, $\mathbf{I}_x$, $\mathbf{I}_y$, $\mathbf{I}_z$ respectively:

$$\check{A} = \begin{bmatrix} a - iv_z & -v_y - iv_x \\ v_y - iv_x & a + iv_z \end{bmatrix}$$

Thus, a rotation $R(\lambda, \mathbf{\Lambda})$ with a quaternion $(\lambda, \mathbf{\Lambda})$ will be represented by the matrix

$$\check{R}^{1/2}(\lambda; \mathbf{\Lambda}) = \begin{bmatrix} \lambda - i\Lambda_z & -\Lambda_y - i\Lambda_x \\ \Lambda_y - i\Lambda_x & \lambda + i\Lambda_x \end{bmatrix}$$

The quaternion $(-\lambda, -\mathbf{\Lambda})$, which corresponds to the same rotation just considered, is mapped by the negative of the matrix $\check{R}^{1/2}(\lambda; \mathbf{\Lambda})$: the isomorphism between Q and SU(2) is thus established. Since SU(2) is the covering group of SO(3), it follows that Q is also the covering group of SO(3). It's useful to notice that $\mathrm{Tr}\check{R}^{1/2}(\lambda; \mathbf{\Lambda}) = 2\lambda$, that is $2cos\frac{1}{2}\phi$, so that binary rotations ($\phi = \pi$) must be represented by traceless matrices as $\mathbf{I}_x$, $\mathbf{I}_y$ and $\mathbf{I}_z$. A binary rotation about an arbitrary axis $r = (x, y, z)$, with $x^2 + y^2 + z^2 = 1$, will be represented by the traceless matrix

$$\check{R}^{1/2}(\pi; \mathbf{n}) = \begin{bmatrix} -iz & -y - ix \\ y - ix & iz \end{bmatrix}$$

This matrix represents the unit quaternion $(0, \mathbf{r})$. Since $\check{R}^{1/2}(\pi; \mathbf{n})$ is a SU(2) matrix, its determinant is equal to 1.

2.2.3 Application: angle, axis of rotation and SU(2) matrices in terms of Euler angles

As an illustration of the power of the quaternion algebra in dealing with rotations, we shall determine the Euler-Rodrigues or quaternion parameters $\lambda = cos\frac{1}{2}\phi$ and $\mathbf{\Lambda} = sin\frac{1}{2}\phi\mathbf{n}$ in terms of Euler angles. $R(\lambda;\mathbf{\Lambda}) = R(\alpha\mathbf{k})R(\beta\mathbf{j})R(\gamma\mathbf{k})$. The quaternion parameters for these rotations can be obtained, and comparing the scalar and vector parts one obtains

$$\begin{aligned}
\lambda &= cos\frac{1}{2}\beta\ cos\frac{1}{2}(\alpha+\gamma)\\
\Lambda_x &= -sin\frac{1}{2}\beta\ sin\frac{1}{2}(\alpha-\gamma)\\
\Lambda_y &= sin\frac{1}{2}\beta\ cos\frac{1}{2}(\alpha-\gamma)\\
\Lambda_z &= cos\frac{1}{2}\beta\ sin\frac{1}{2}(\alpha+\gamma)
\end{aligned}$$

It is now trivial to obtain ϕ and $\mathbf{n}$ in terms of Euler angles:

$$\begin{aligned}
cos\frac{1}{2}\phi &= cos\frac{1}{2}\beta\ cos\frac{1}{2}(\alpha+\gamma)\\
n_x &= -\left(sin\frac{1}{2}\phi\right)^{-1} sin\frac{1}{2}\beta\ sin\frac{1}{2}(\alpha-\gamma)\\
n_y &= \left(sin\frac{1}{2}\phi\right)^{-1} sin\frac{1}{2}\beta\ cos\frac{1}{2}(\alpha-\gamma)\\
n_z &= \left(sin\frac{1}{2}\phi\right)^{-1} cos\frac{1}{2}\beta\ sin\frac{1}{2}(\alpha+\gamma)
\end{aligned}$$

Because $-\pi < \phi \leq \pi$, $cos\frac{1}{2}\phi$ must always be ≥ 0, but that $sin\frac{1}{2}\phi$ is undetermined as to sign, whence $\mathbf{n}$ is also undetermined as to sign. Thus, the Euler angles cannot distinguish between $R(\phi\mathbf{n})$ and $R(-\phi,-\mathbf{n})$. Even if these rotations are the same, a distinction between them (i.e. between their poles) is necessary in order to satisfy the continuity conditions in SO(3). It is thus most important to realize that such continuity conditions in SO(3) aren't necessarily satisfied when Euler angles are used.

The corresponding matrix in terms of Euler angles would be

$$\check{R}^{1/2}(\alpha\beta\gamma) = \begin{bmatrix} cos\frac{1}{2}\beta^{-\frac{1}{2}i(\alpha+\gamma)} & -sin\frac{1}{2}\beta^{-\frac{1}{2}i(\alpha-\gamma)} \\ sin\frac{1}{2}\beta^{\frac{1}{2}i(\alpha-\gamma)} & cos\frac{1}{2}\beta^{\frac{1}{2}i(\alpha+\gamma)} \end{bmatrix}$$

Chapter 3

Quaternions

In mathematics, quaternions are a number system that extends the complex numbers set. They were formalized in 1843 by the Irish mathematician Sir William Rowan Hamilton, who was looking for a method to extend complex numbers, which can be seen as points on a plane, on a bigger number of spacial dimensions. He tried at first with a tridimensional extension, without success: then, he realized that a 4-dimensional extension was needed: the quaternions [13].
Today, quaternions are used in rotation representations in 3D space, above all in 3D computer graphics. The reason is that the combination of many transformations described by quaternions is numerically more stabe than the combination of many matrix transformations. The representation of a rotation as a quaternion (4 numbers) is more compact than the representation as an orthogonal matrix (9 numbers). Furthermore, for a given axis and angle, one can easily construct the corresponding quaternion, and conversely, for a given quaternion one can easily read off the axis and the angle. Both of these are much harder with matrices or Euler angles.
Quaternions also avoid a phenomenon called gimbal lock, which is the loss of one degree of freedom in a three-dimensional, three-gimbal mechanism. It can result when, for example. in roll-pitch-yaw rotational systems, the pitch is rotated 90° up or down, so that yaw and roll then correspond to the same motion, and a degree of freedom of rotation is lost. In general, it occurs when the axes of two of the three gimbals are driven into a parallel configuration, "locking" the system into rotation in a degenerate two-dimensional space.

3.1 Three equivalent definitions of quaternion

A quaternion q is given by the sum of an ordered pair made up of a scalar component s and a 3D vectorial component $\mathbf{v}$:

$$q = (s, \mathbf{v}) = s + \mathbf{v}$$

Another way to call $\mathbf{v}$ is "imaginary component", because it's a vector with three complex components.
The set of quaternions is indicated with $\mathbb{H}$. A quaternion q belonging to $\mathbb{H}$ may be defined through a linear combination of elements 1, i, j, k:

$$q = s + ai + bj + ck$$

where $s \in \mathbb{R}$ and $(a, b, c) \in \mathbb{R}$ are the components of $\mathbf{v}$. Moreover, literal symbols i, j, k satisfy the following properties:

$$i^2 = j^2 = k^2 = -1$$

$$ij = -ji = k$$

$$jk = -kj = i$$

$$ki = -ik = j$$

Therefore:

$$ijk = -1$$

i, j, k are multiplied following the same rules of a vector product between unit vectors of coordinate axes of a right-hand reference system.
The last way to indicate a quaternion q is by means of a quadruple of real numbers:

$$q = (s, a, b, c)$$

This is possible thanks to the analogy with complex numbers, where $s + ai$ is represented by the real couple (s, a).

3.1.1 Considerations

Quaternions are a general mathematical entity which includes:

- real numbers: $s = (s, 0, 0, 0)$, $s \in \mathbb{R}$
- complex numbers: $s + ai = (s, a, 0, 0)$, $s, a \in \mathbb{R}$

- vectors in $\mathbb{R}^3$: $\mathbf{v} = (0, a, b, c)$, $a, b, c \in \mathbb{R}$; in this case, however, i, j, k must be interpreted as unit vectors of coordinate axes of a right-hand reference system.

A quaternion of the form $(s, \mathbf{0})$ is called a real quaternion. Because they multiply precisely like real numbers, $(s, \mathbf{0})(t, \mathbf{0}) = (st, \mathbf{0})$; they can be identified with real numbers: $(s, \mathbf{0}) = s$. We can, in this way, define the product between a quaternion q and a scalar $\alpha \in \mathbb{R}$, which is known as external product:

$$\alpha q = (\alpha s, \alpha \mathbf{v})$$

A quaternion of the form $(0, \mathbf{v})$ is called a pure quaternion. If the quaternion is pure,

$$q_1 q_2 = (-\mathbf{v}_1 \cdot \mathbf{v}_2, \mathbf{v}_1 \wedge \mathbf{v}_2)$$

3.2 Algebra of quaternions

3.2.1 Sum of quaternions

Given two quaternions q_1 and q_2,

$$q_1 + q_2 = (s_1 + s_2, \mathbf{v}_1 + \mathbf{v}_2)$$

The quaternion $(0, 0, 0, 0) = (0, \mathbf{0})$ is the neutral element of addition. The algebric structure $(\mathbb{H}, +)$ is an abelian group.

3.2.2 Product of quaternions

Given $q_1 = (s_1, \mathbf{v}_1) = s_1 + a_1 i + b_1 j + c_1 k$ and $q_2 = (s_2, \mathbf{v}_2) = s_2 + a_2 i + b_2 j + c_2 k$,

$$q_1 q_2 = (s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2, s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v}_1 \wedge \mathbf{v}_2)$$

where $\mathbf{v}_1 \cdot \mathbf{v}_2 = a_1 a_2 + b_1 b_2 + c_1 c_2 \in \mathbb{R}$ is a scalar product, while
$\mathbf{v}_1 \wedge \mathbf{v}_2 = (b_1 c_2 - b_2 c_1) i - (a_1 c_2 - a_2 c_1) j + (a_1 b_2 - a_2 b_1) k$ is a vector product.
Product between quaternions is associative (i.e., $q_1(q_2 q_3) = (q_1 q_2) q_3$), but it's not commutative (i.e., $q_1 q_2 \neq q_2 q_1$); it's also distributive.
The quaternion $(1, 0, 0, 0) = (1, \mathbf{0})$ is the neutral element of multiplication.

3.2.3 Algebric structure

The set of quaternions, with operations of sum and product between quaternions, establish a non cummutative with respect to product. The set of quaternions, with operations of sum

between quaternions and product with a scalar, establish a real vector space of dimension 4. A base for this space is given by $(1, i, j, k)$.

3.3 Conjugate and Norm

Given a generic quaternion q, its conjugate would be

$$\bar{q} = (s, -\mathbf{v}) = s - ai - bj - ck$$

If q is a pure quaternion, then $\bar{q} = -q$. The properties of the conjugate are:

- $\bar{\bar{q}} = q$
- $\overline{q_1 q_2} = \bar{q}_2 \bar{q}_1$
- $\overline{q_1 + q_2} = \bar{q}_1 + \bar{q}_2$

Given a generic quaternion q, its norm would be

$$|q| = \sqrt{q\bar{q}} = \sqrt{\bar{q}q} = \sqrt{s^2 + \mathbf{v} \cdot \mathbf{v}} = \sqrt{s^2 + a^2 + b^2 + c^2}$$

The norm of q it's always a positive real number, it would be null only if $q = 0$. The properties of the norm and of the inverse are:

- $|q|^2 = q\bar{q} = \bar{q}q$
- $|q_1 q_2| = |q_1||q_2|$
- $q^{-1} = \dfrac{\bar{q}}{|q|^2}$
- $|q^{-1}| = \dfrac{1}{|q|} = |q|^{-1}$
- $\overline{q^{-1}} = (\bar{q})^{-1}$
- $(q_1 q_2)^{-1} = q_2^{-1} q_1^{-1}$

3.4 Exponential, logarithm and power functions

Given a generic quaternion $q = a + \mathbf{v}$, the exponential is computed as

$$\exp(q) = \sum_{n=0}^{\infty} \frac{q^n}{n!} = e^a \left(\cos \|\mathbf{v}\| + \frac{\mathbf{v}}{\|\mathbf{v}\|} \sin \|\mathbf{v}\| \right)$$

while the logarithm is

$$\ln(q) = \ln \|q\| + \frac{\mathbf{v}}{\|\mathbf{v}\|} \arccos \frac{a}{\|q\|}$$

The power of q raised to an arbitrary real exponent α is given by:

$$q^\alpha = \|q\|^\alpha e^{\hat{n}\alpha\theta} = \|q\|^\alpha \left(\cos(\alpha\theta) + \hat{n}\sin(\alpha\theta)\right)$$

3.5 Unit quaternions

Unit quaternions are those quaternions whose norm is 1. The set of unit quaternions is a 3-dimensional hypersphere in a 4D space:

$$S^3 = \{(s, a, b, c) \in \mathbb{R}^4 | s^2 + a^2 + b^2 + c^2 = 1\}$$

If q is a unit quaternion, $q^{-1} = \bar{q}$.
Unit quaternions form a non abelian multiplicative group with respect to product.

3.6 Quaternions and Rotations

Each rotation of a point in three dimensions can be represented by an axis and a rotation angle: unit quaternions are an easy tool to code with four numbers these informations axis-angle. Rotation would be applied to a position vector, that is, a vector which represents the position of a point with respect to the origin of the reference system in $\mathbb{R}^3$.
We want to rotate a point $P \in \mathbb{R}^3$ by an angle θ (in radians), around a rotation axis which passes through the origin of a right-hand reference system. The rotation axis would be identified by an unit vector $\mathbf{n} = (n_x, n_y, n_z)$. From this, the following unit quaternion is defined:

$$q = (s, \mathbf{v}) = cos\frac{\theta}{2} + (n_x i + n_y j + n_z k) sin\frac{\theta}{2} = \left(cos\frac{\theta}{2}, \mathbf{n} sin\frac{\theta}{2}\right)$$

which is a quaternion in its polar form. Point $P \in \mathbb{R}^3$ would be identified by the vector $\mathbf{r} = (p_1 p_2 p_3)$; the pure quaternion which would represent P is $p = (0, \mathbf{r}) = (0, p_1, p_2, p_3)$. In this way, points in $\mathbb{R}^3$ are identified with quaternions with first coordinate null.
Rotation determined by q and applied to p is given by the following conjugate operation:

$$q \longmapsto qpq^{-1}$$

The result of this operation is the vector $p' = (0, \mathbf{r'})$, which belongs to the set of pure quaternions on $\mathbb{R}^3$. Each map defined in this way is indeed a rotation, because it preserves the norm:

$|qpq^{-1}| = |q||p||q^{-1}| = |q||p||q|^{-1} = |p|$.

Vector p' is equal to vector p rotated by an angle θ around the axis through the origin, indicated by unit vector $\mathbf{n}$. The scalar component of p' is null, while the vectorial component is given by $\mathbf{r'} = (cos\theta)\mathbf{r} + (1 - cos\theta)\mathbf{n}(\mathbf{n} \cdot \mathbf{r}) + (sin\theta)(\mathbf{n} \wedge \mathbf{r})$. Again, this rotation is around an axis through the origin: for any other rotations, first we need to translate the axis into the origin, then make the rotation and eventually translate again the axis in the original position.

A sequence of rotation would be given by a sequence of quaternions, in particular the composition of two subsequent rotations can be obtained multipling the two corresponding unit quaternions, taking care of the order (because the product isn't commutative) and of the fact that the two quaternions have to be referred to the same rotation axis.

The inverse of a rotation is given by the inverse of the corresponding quaternion, which is the conjugate in case of unit quaternions.

If we write $q = [s, l, m, n]'$ and $\mathbf{v} = [l, m, n]'$, it's easy to see that

$$qpq^{-1} = (0, 2(\mathbf{v}, p)\mathbf{v} + (s^2 - \|\mathbf{v}\|^2)p + 2s\mathbf{v} \wedge p)$$

We can also write $\mathbf{v} \wedge p = \tilde{\mathbf{v}}p$, where

$$\tilde{\mathbf{v}} = \begin{bmatrix} 0 & -n & m \\ n & 0 & -l \\ -m & l & 0 \end{bmatrix}$$

and $(\mathbf{v}.p)\mathbf{v} = Ap$, where A is a 3x3 matrix given by

$$A = [l\mathbf{v}, m\mathbf{v}, n\mathbf{v}] = \begin{bmatrix} l^2 & lm & ln \\ lm & m^2 & mn \\ ln & mn & n^2 \end{bmatrix}$$

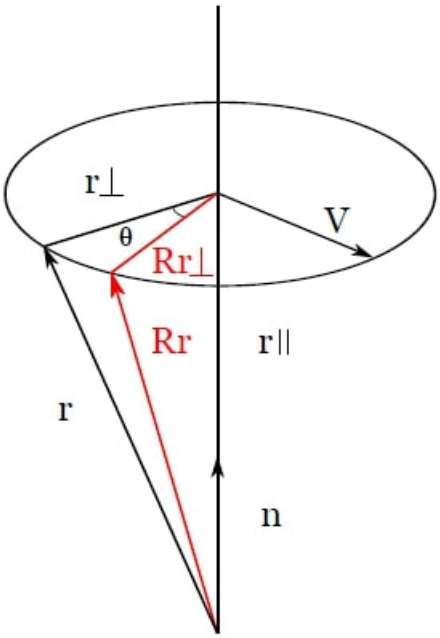
Therefore, the corresponding rotation matrix R would be

$$R = \begin{bmatrix} s^2 + l^2 - m^2 - n^2 & 2(lm - sn) & 2(ln + sm) \\ 2(lm + sn) & s^2 - l^2 + m^2 - n^2 & 2(mn - sl) \\ 2(ln - sm) & 2(mn + sl) & s^2 - l^2 - m^2 + n^2 \end{bmatrix}$$

It appears that the coefficients of R are polynomial functions of the coordinates of q. The relation $s^2 + l^2 + m^2 + n^2 = 1$ must be always satisfied.

3.7 Angular displacement

The same formula could be achieved by taking on exam a rotation without the use of quaternions: we want to rotate vector $\mathbf{r}$ around axis $\mathbf{n}$ by an angle θ, as in figure.



Vector $\mathbf{r}$ may be decomposed in a parallel component to $\mathbf{n}$, $\mathbf{r} \parallel = (\mathbf{n} \cdot \mathbf{r})\mathbf{n}$, and an orthogonal component to $\mathbf{n}$, $\mathbf{r} \perp = \mathbf{r} - \mathbf{r} \parallel$. The parallel component remains unchanged during rotation, and after the rotation is denoted by $\mathbf{Rr} \parallel$.
The orthogonal component instead changes, and after the rotation will be $\mathbf{Rr} \perp$.
Given the vector $\mathbf{V}$ othogonal to $\mathbf{r} \perp$, it would be $\mathbf{V} = \mathbf{n} \wedge \mathbf{r} \perp = \mathbf{n} \wedge \mathbf{r}$, then $\mathbf{Rr} \perp$ could be expressed as a function of $\mathbf{V}$: $\mathbf{Rr} \perp = (cos\theta)\mathbf{r} \perp +(sin\theta)\mathbf{V}$. Therefore

$$\mathbf{Rr} = \mathbf{Rr} \parallel +\mathbf{Rr} \perp = (cos\theta)\mathbf{r} + (1 - cos\theta)\mathbf{n}(\mathbf{n} \cdot \mathbf{r}) + (sin\theta)(\mathbf{n} \wedge \mathbf{r})$$

3.8 Quaternions and matrices

We know that a rigid rotation in $\mathbf{R}^3$ may be represented by a rotation matrix 3x3, which is orthogonal and with unitary determinant. We can associate a rotation matrix to each quaternion and viceversa; the map would be surjective, because both q and $-q$ cause the same rotation.
In order to rotate a vector $\mathbf{p}$ by an angle θ with the quaternion q, we use the conjugate operator qpq^{-1}, where $q = \left(cos\frac{\theta}{2}, \mathbf{n}sin\frac{\theta}{2}\right) = (w, x, y, z)$. It can be proved that the same operation could be fulfilled by applying to vector $(\mathbf{p}, 0)$ the rotation matrix

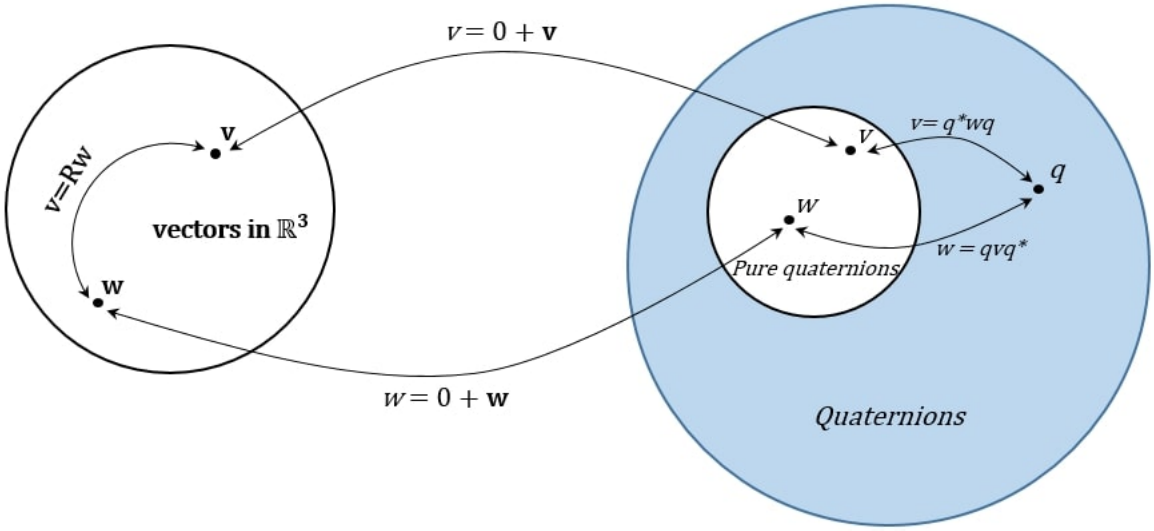
$$M = \begin{bmatrix} 1 - 2(y^2 + z^2) & 2xy - 2wz & 2wy + 2xz & 0 \\ 2xy + 2wz & 1 - 2(x^2 + z^2) & -2wx + 2yz & 0 \\ -2wy + 2xz & 2wx + 2yz & 1 - 2(x^2 + y^2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The inverse transformation from matrix to quaternion, consists in taking a generic matrix $\{M_{i,j}\}$ for $i, j = 0, 1, 2, 3$, with the following characteristics:

- $M_{3,3} = 1$;
- $M_{0,3} = M_{1,3} = M_{2,3} = M_{3,0} = M_{3,1} = M_{3,2} = 0$;
- Tr $M = 4 - 4(x^2 + y^2 + z^2)$;
- $w^2 + x^2 + y^2 + z^2 = 1, \implies$ Tr $M = 4w^2$.

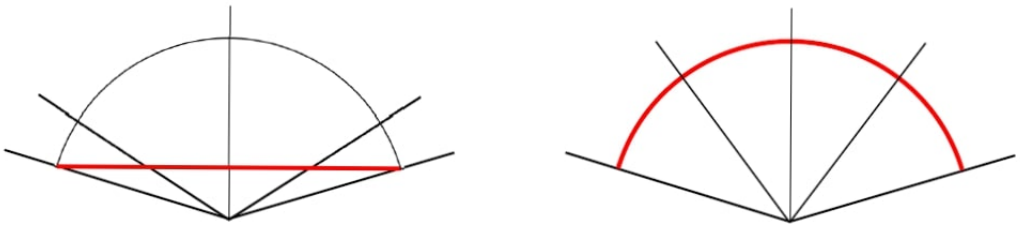
Thanks to this, it follows that

$$\begin{aligned} w &= \pm\frac{1}{2}\sqrt{M_{0,0} + M_{1,1} + M_{2,2} + M_{3,3}} \\ x &= \frac{M_{2,1} - M_{1,2}}{4w} \\ y &= \frac{M_{0,2} - M_{2,0}}{4w} \\ z &= \frac{M_{1,0} - M_{0,1}}{4w} \end{aligned}$$

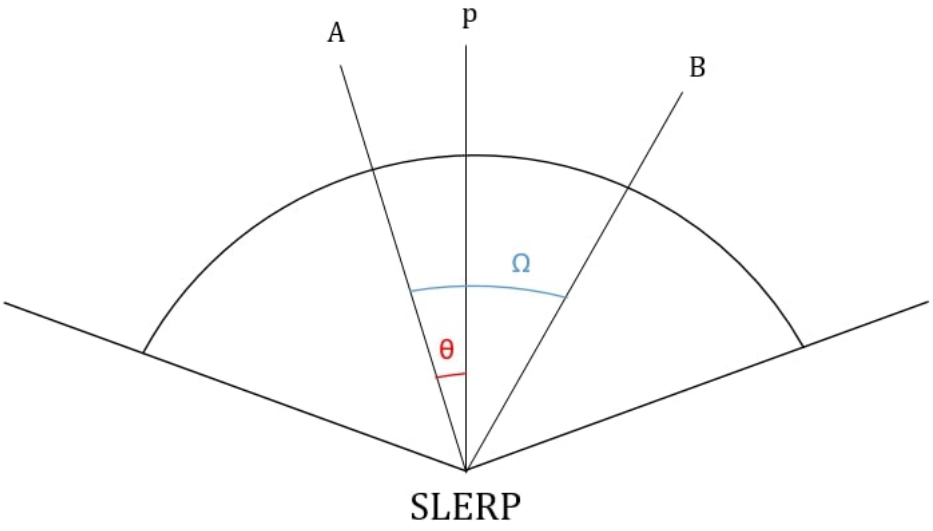


3.9 SLERP - spherical linear interpolation

In order to interpolate two unit quaternions and obtain intermediate quaternions which identify rotation matrices, it has to be kept in mind that the space of unit quaternions consists of an hypersphere in four dimensions, then all quaternions obtained through the interpolation lie on the hypersphere.



A linear interpolation of two quaternions would cause different angles and thus a velocity variation (see figure above, LERP on le left and SLERP on the right). Then we need to introduce the spherical interpolation: we interpolate along a geodesic line which has the extremes in key points. The geodesic line is the arc of circumference identified by the intersection between the sphere and a plane passing through the two key points and the origin. The displacement on the arc occurs at constant velocity and would depend on a parameter set between 0 and 1.
For simplicity, let's take in exam the situation in two dimensions (following figure): let A and B be the first and the last point of the arc, separated by an angle Ω. Let p be the intermediate point, which forms with A an angle θ. It will be: $cos\Omega = A \cdot B$, scalar product between unit vectors which go from the origin to the arc edge. Moreover, $cos\theta = A \cdot p$ and $|p| = 1$. SLERP is based on the fact that each point of the curve must be a linear combination of the edges.



Point p will be given by the parametric equation

$$p = A\frac{sin(\Omega - \theta)}{sin\Omega} + B\frac{sin\theta}{sin\Omega}$$

Generalizing in 4D the interpolation between two unit quaternions q_1 and q_2 forming the angle $q_1 \cdot q_2 = cos\Omega$, considering $\theta = u$ as a parameter varying between 0 and 1, it would be

$$SLERP(q_1, q_2, u) = q_1\frac{sin((1-u)\Omega)}{sin\Omega} + q_2\frac{sin(u\Omega)}{sin\Omega}$$

There exist two possible geodesic arcs from q_1 to q_2: one which follows the shortest path, the other which follows the longest, and this means to interpolate along angle Ω or along angle $2\pi - \Omega$. This is possible because conjugate operation produces the same result with both q and $-q$. In order to decide which path should be followed, we have to calculate $(q_1 - q_2) \cdot (q_1 - q_2)$ and $(q_1 + q_2) \cdot (q_1 + q_2)$ and choose the lower value, which is necessary to decide whether to substitute q_2 with $-q_2$.

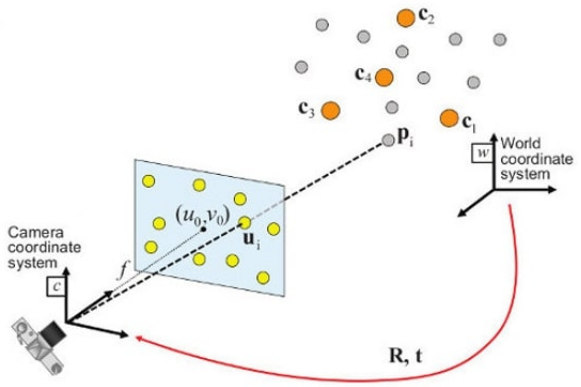
Chapter 4

Pose estimation

In robotics, the identification of a specific rigid body (3D) in an image (2D) and the determination of its position and orientation (with respect to a prefixed frame) is a common task. Thanks to this information, a robot can manipulate the rigid body, or in other cases, it can avoid moving against the rigid body. The pose of a rigid body is given by the combination of its position and orientation, even if this concept is sometimes used only to describe the orientation. However, there exist environments where corners and edges are difficult to extract from an image. Trying to avoid this issue, the body is dealt with as a whole in noted techniques through the use of free-form contours.

This problem generally requires estimating six degrees of freedom of the pose and some calibration parameters as the focal length, the principal point, the aspect ratio and the skew.

The most common simplification is to assume known calibration parameters which is the so-called Perspective-*n*-Point problem:



Given a set of correspondences between 3D points p_i expressed in a world reference frame, and

their 2D projections u_i onto the image, we seek to retrieve the pose (R and t) of the camera with respect to the world and the focal length f. In particular, we try to develop fast pose estimation algorithms which produce stable results for a small number of point or line correspondences.
There are four pose estimation problems with point data [16]; each of them arises from two views taken of the same object that can be thought of as having undergone an unknown rigid body motion from the first view to the second view. One "view" provides 3D data relative to the model reference frame, the other is the 2-D perspective projection.
In the simplest pose estimation problem, the data sets consist of 2D data points in a 2D space. Such data sets arise naturally when flat 3D rigid bodies are viewed under perspective projection with the look angle being the same as the surface normal of the object viewed. In the next more difficult pose estimation problem, the data sets consist of 3D data points in a 3D space. Such data sets arise naturally when 3D objects are viewed with a range finder sensor. In the most difficult pose estimation problems, one data set consists of the 2D perspective projection of 3D points and the other data set consists of either a 3D point data set, in which case it is known as absolute orientation problem, or the other data set consists of a second 2D perspective projection view of the same 3D point data set, in which case it's known as the relative orientation problem. The latter case occurs with time-varying imagery, uncontrolled stereo or multicamera imagery.

4.1 2D - 2D estimation

The first task to be accomplished is determining the relationship between the coordinate system of the observed image and the coordinate system of the model. In this way, it is possible to determine whether each device needed is present and whether everything observed to be present is actually present and in its correct position and orientation. Usually, the relationship between the two coordinate systems is given by a 2D rotation and translation.
During the matching process, the noise may disturbe the pose estimation, causing an incorrect match: a data point of the model would then correspond to an incorrect point of the image. These incorrect points, which are called outliers, may affect the accuracy and stability of the pose estimation. Therefore, a new robust method to weaken the effect of the outliers is needed; moreover, it would improve the performance and reliability of the least-squares method.
So, let us discuss first the least-squares method and then the robust method.

4.1.1 Problem

In the simple 2D pose detection problem, we are given N two-dimensional coordinate observations from the observed image: $x_1, \cdots, x_N$. These could correspond, for example, to the observed center position of all observed rigid bodies. We are also given the corresponding or matchng N 2D coordinate vectors from the model: $y_1, \cdots, y_N$. There is an approximate rotational and translational relationship between the image coordinate system and the rigid body coordinate system: it allows an easy and quick matching, just by coupling a rotated and translated image position to a rigid body position. The match is established if the rotated image position is close enough to the one of the rigid body.

Considering the ideal case, the 2D pose detection problem could be carried out by determining, from the matched points, a more precise estimate of a rotation matrix R and a translation t that minimize the weighted sum of the residual errors ϵ^2:

$$\epsilon^2 = \sum_{n=1}^{N} w_n \|y_n - (Rx_n + t)\|^2$$

The weights w_n, $n = 1, \cdots, N$ satisfy $w_n \geq 0$ and $\sum_{n=1}^{N} w_n = 1$.

4.1.2 Least-squares method

Expanding the espression of the residual errors, we'll obtain

$$\epsilon^2 = \sum_{n=1}^{N} w_n[(y_n - t)'(y_n - t) - (y_n - t)'Rx_n - x_n'R'(y_n - t) + x_n'R'Rx_n]$$

As a rotation matrix, R is orthonormal, then $R^{-1} = R'$. Plus, since $(y_n - t)'Rx_n$ is a scalar, it's equal to its transpose.

Taking the partial derivative of the new ϵ^2 with respect to the components of the translation t, and setting the partial derivative to 0, we obtain

$$0 = \sum_{n=1}^{N} w_n[-2(y_n - t) + 2Rx_n]$$

Considering $\bar{x} = \sum_{n=1}^{N} w_n x_n$ and $\bar{y} = \sum_{n=1}^{N} w_n y_n$, we will obtain

$$\bar{y} = R\bar{x} + t$$

We can then simplify the expression of the residual error by substituting $t = \bar{y} - R\bar{x}$.

The counterclockwise rotation angle θ is related to the rotation matrix by

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

First, let us take the partial derivative of ϵ^2 with respect to θ; then, after some calculations, we will obtain an equation like

$$0 = A\sin\theta + B\cos\theta$$

where

$$A = \sum_{n=1}^{N} w_n[(y_{n1} - \bar{y}_1)(x_{n1} - \bar{x}_1) + (y_{n2} - \bar{y}_2)(x_{n2} - \bar{x}_2)]$$

$$B = \sum_{n=1}^{N} w_n[(y_{n1} - \bar{y}_1)(x_{n2} - \bar{x}_2) - (y_{n2} - \bar{y}_2)(x_{n1} - \bar{x}_1)]$$

Therefore we can state that

$$\cos\theta = -\frac{A}{\sqrt{A^2 + B^2}} \qquad \sin\theta = \frac{B}{\sqrt{A^2 + B^2}}$$

or

$$\cos\theta = \frac{A}{\sqrt{A^2 + B^2}} \qquad \sin\theta = -\frac{B}{\sqrt{A^2 + B^2}}$$

The correct value for θ will in general be unique and will be that θ that minimizes ϵ^2. By substituting each value for θ into the original expression, we can provide the best solution choice.

In this subsection, w_n has always been assumed to be given. But if we want to improve the performance and the stability of pose estimation, the weights need to be determined based on the data. Therefore, we need a method to assign a weight based on the residual error. The method to assign appropriate weights to the data points is done by a robust method using an iterative weighted least-squares method, which is described in the next subsection.

4.1.3 Robust method

In this subsection we will introduce an iterative weighted least-squares method where the weights are data dependent. The purpose is to make the outliers have zero or small weights and thus to eliminate the effects of them in the pose estimation.

M-estimator

In order to find the solution for θ, we have to minimize the problem in the following way:

$$min_\theta \sum_{i=1}^{N} \rho(x_i - \theta)$$

Using an implicit equation, we will have

$$\sum_{i=0}^{N} \psi(x_i - \theta) = 0$$

where N is the sample size and ρ is an arbitrary nonnegative monotonically increasing function (called the object function) for positive argument and monotonically decreasing for negative argument.
Moreover, $\psi(x_i - \theta)$ is called M-estimator and it's equal to

$$\psi(x_i - \theta) = \frac{\partial}{\partial\theta}\rho(x_i - \theta)$$

If we want to represent θ as a weighted mean, we should consider $w_i = \dfrac{\psi(x_i - \theta)}{(x_i - \theta)}$ where $i = 1, \cdots, N$, and then

$$\theta = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{n} w_i}$$

where w_i depend on data.

Iterative weighted least-squares method

The residual error ϵ_i for nth data sample is $\epsilon_i = y_i - (Rx_i + t)$.
Given the data sets x_i and y_i, where $i = 1, \cdots, N$,the robust estimation procedure is implemented as the following iterative method.

1. Select initial starting values for R and t.

2. To find the weights given by R and t, we use the Turkey's biweight function

$$w_i = \begin{cases} \left[1 - \left(\dfrac{x}{cS}\right)^2\right]^2 & |x| \leq cS \\ 0 & otherwise \end{cases}$$

   where x^2 is replaced by the residual error:

$$w_i = \begin{cases} \left[1 - \dfrac{\|\epsilon_i\|^2}{(cS)^2}\right]^2 & \|\epsilon_i\| \leq cS \\ 0 & otherwise \end{cases}$$

   The new R and t are obtained from the new weights.

3. If some degree of convergence in R and t are obtained, go to step 4. If not, go back to step 2.

4. From the final W, we normalize the weights and find estimates for rotation angle and translation using the solution derived in the past Subsection.

4.2 3D-3D estimation

4.2.1 Problem

Consider a rotation matrix R and a translation vector t; then, let $x_1, \cdots, x_n$ be the points in Euclidean 3D space that match the points $y_1, \cdots, y_n$ belonging to the same space: each x_n is the same rigid body motion of y_n.
In particular, $y_n = Rx_n + t + \eta_n$, where η_n represents the noise.
The problem of 3D-3D pose estimation is to infer R and t from $x_1, \cdots, x_n$ and $y_1, \cdots, y_n$.

4.2.2 Derivation

To determine R and t we set up a constrained least-squares problem.
First of all, we minimize

$$\sum_{n=1}^{N} w_n \|y_n - (Rx_n + t)\|^2$$

The constraint is that $R' = R^{-1}$, since R is a rotation matrix. Letting

$$R = \begin{bmatrix} r_1' \\ r_2' \\ r_3' \end{bmatrix}$$

where each r_i is a 3x1 vector, the constraints would amount to six equations:
$r_1' r_1 = r_2' r_2 = r_3' r_3 = 1$ and $r_1' r_2 = r_1' r_3 = r_2' r_3 = 0$.
The least-squares problem with these constraints can be written minimizing the residual error ϵ^2, then taking the partial derivative with respect to t_n and setting it equal to zero:

$$\sum_{n=1}^{N} w_n (y_n - Rx_n - t) = 0$$

By rearranging we obtain $t = \bar{y} - R\bar{x}$, where

$$\bar{x} = \frac{\sum_{n=1}^{N} w_n x_n}{\sum_{n=1}^{n} w_n} \qquad \bar{y} = \frac{\sum_{n=1}^{N} w_n y_n}{\sum_{n=1}^{n} w_n}$$

Now we substitute $\bar{x} - R\bar{y}$ in the definition of ϵ^2, and then we take the partial derivative of it with respect to the components of each y_n. Setting these partial derivative to zero and rearranging, we obtain an equation in the form $AR' + R'\Lambda = B$, where

$$A = \sum_{n=1}^{N} (x_n - \bar{x})(x_n - \bar{x})'$$

$$\Lambda = \begin{bmatrix} \lambda_1 & \lambda_4 & \lambda_5 \\ \lambda_4 & \lambda_2 & \lambda_6 \\ \lambda_5 & \lambda_6 & \lambda_3 \end{bmatrix}$$

$$B = (b_1 b_2 b_3) \qquad where \qquad b_k = \sum_{n=1}^{N} w_n (y_{nk} - \bar{y}_k)(x_n - \bar{x})$$

Rearranging the equation we obtain $RB = (RB)'$, and then it's easy to find the solution for R. Let $B = UDV$ where U and V are orthonormal and D is diagonal. then $R = V'U'$.

4.3 2D perspective projection - 3D pose estimation

Consider a rotation matrix R and a translation vector t; then, let $y_1, \cdots, y_n$ be the observed 3D model points in Euclidean space, and let (u_{n1}, u_{n2}) $(n = 1, \cdots, N)$ be the corresponding 2D perspective projection of the 3D model points. The relationship is given by

$$u_{n1} = f\frac{r_1 y_n + t_1}{r_3 y_n + t_3}$$

$$u_{n2} = f\frac{r_2 y_n + t_2}{r_3 y_n + t_3}$$

$$t = (t_1, t_2, t_3)'$$

$$R = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}$$

f is the focal length, which is the distance of the image plane in front of the origin that is the center of perspectivity.

Given the 3D model points and the corresponding 2D perspective projection points on the image plane, by determining R and t we would manage to solve the problem of pose estimation.

4.3.1 Iterative least-squares solution

Then, let us see iterative procedures for determining a least-squares solution for R and t. In the following, we use the superscript or subscript k to denote the values in the kth iteration step. Moreover, let

$$x_n = \begin{bmatrix} x_{n1} \\ x_{n2} \\ x_{n3} \end{bmatrix} = R \begin{bmatrix} y_{n1} \\ y_{n2} \\ y_{n3} \end{bmatrix} + t$$

be the rotated and translated point of y_n. Eventually, let d_n be the estimated depth of each point x_n relative to the camera coordinate system.

Method 1

1. Choose initial reasonable values for the depth d_n^0 of each point.

2. Iterate. Suppose that the depth values are given. Define the depth values for the $(k+1)$th iteration by:

   (a) Find the rotation matrix R_k and the translation vector t_k that minimize

   $$\epsilon_k^2 = \sum_{n=1}^{N} w_n \| R_k y_n + t_k - d_n^k v_n \|^2$$

   R_k and t_k are the solution of the pose estimation problem.

   (b) Define

   $$d_n^{k+1} = \left(\frac{D_y}{D_x} \right) x_{n3}^k$$

   where

   $$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_n \qquad \bar{y} = \frac{1}{N} \sum_{n=1}^{N} y_n \qquad D_y = \sum_{n=1}^{N} \| y_n - \bar{y} \|^2 \qquad D_x = \sum_{n=1}^{N} \| x_n - \bar{x} \|^2$$

Method 2

Replace step 2b of Method 1 with the following considerations: define d_n^{k+1} by

$$d_n^{k+1} = \frac{(R_k + t_k)' v_n}{v_n' v_n}$$

It can be demonstrated that $\epsilon_{k+1}^2 \leq \epsilon_k^2$.

4.3.2 Robust M-estimation

An M-estimate is any T_k defined by a minimization problem of the form

$$min \sum_{i=1}^{n} \rho(x_i - T_k)$$

or by an implicit equation

$$\sum_{i=1}^{n} \psi(x_i - T_k) = 0$$

where ρ is an arbitrary function (called object function), $\psi(x_i - T_k) = \frac{\partial}{\partial T_k} \rho(x - T_k)$, and

$$T_k = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

In robust estimation, the estimates are obtained only after an iterative process because the estimates do not have closed forms; here are exposed two different iterative methods.

Modified residual method

Before solving the problem, the residuals are modified by a proper ψ function; the aim of this iterative procedure is to find θ.

1. Choose an initial approximation θ^0.
2. Iterate. Given the estimation θ^k in step k, compute the solution in the $(k+1)$th step as follows:
   (a) Compute the modified residuals
   $$r_i^* = \psi\left(\frac{r_i}{S^k}\right) S^k$$
   where
   $$r_i = y_i - f_i(\theta^k) \qquad S^k = \frac{median|r_i|}{0.6745} \qquad r_i \neq 0, \quad i = 1, \cdots, n$$
   (b) Solve the least-squares problem $X\delta = r^*$, where $X = [x_{ij}]$ is the gradient matrix as
   $$x_{ij}\frac{\partial}{\partial\theta_j} f_i(\theta^k)$$
   The solution for this equation can be found using the standard least-squares method.
   (c) Set $\theta^{k+1} = \theta^k + \hat{\delta}$.

Modified weights method

If we take the partial derivative of ρ with respect to θ and we set it to zero, we will get, in the standard weighted form,

$$\sum_{i=1} w_i r_i \frac{\partial f_i(\theta)}{\theta_j} = 0 \qquad where \quad w_i = \frac{\psi\left(\frac{r_i}{S}\right)}{\left(\frac{r_i}{S}\right)}$$

Therefore the iterative procedure to determine θ is as follows:

1. Choose an initial approximation θ^0.
2. Iterate. Given θ^k at the kth step, compute θ^{k+1} as follows:
   (a) Solve $PX\delta = Pr$ where
   $$P = \begin{bmatrix} \sqrt{w_1} & & \\ & \ddots & \\ & & \sqrt{w_n} \end{bmatrix}$$
   (b) If $\hat{\delta}$ is the solution of step 2a, then set $\theta^{k+1} = \theta^k + \hat{\delta}$.

4.3.3 Pseudocode for extracting 3D from 2D

The algorithm for determining pose estimation is based on the iterative closest point algorithm. The main idea is to determine the correspondences between 2D image features and points on the 3D model curve.

1. Reconstruct projection rays from the image points;
2. Estimate the nearest point of each projection ray to a point on the 3D contour;
3. Estimate the pose of the contour with the use of this correspondence set;
4. Go to 2.

If the rigid body is partially occluded, this algorithm won't work. Therefore, we need a stronger algorithm which assumes that all contours are rigidly coupled, i.e. the pose of one contour defines the pose of another contour:

1. Reconstruct projection rays from the image points;
2. For each projection ray R:
3. For each 3D contour:
   (a) Estimate the nearest point P_1 of ray R to a point on the contour;
   (b) if (n==1) choose P_1 as actual P for the point-line correspondence;
   (c) else compare P_1 with P:
   if $dist(P_1, R)$ is smaller than $dist(P, R)$, then choose P_1 as new P
4. Use (P, R) as correspondence set;
5. Estimate pose with this correspondence set;
6. Transform contours, go to 2.

4.4 Three point perspective pose estimation problem

Thanks to its wide variety of applications, a very common problem in photogrammetry and computer vision is the three point space resection problem. In this section, different solutions

given through history will be analyzed, from the first half of XIX century to the more recent computer vision of late XX century.

The problem could be stated as follows [17] : "given the perspective projection of three points constituting the vertices of a known triangle in 3D space, determine the position of each of the vertices.". There may be as many as four possible solutions for point positions in front of the center of perspectivity and four corresponding solutions whose point positions are behind the center of perspectivity.

4.4.1 Definition of the problem

The first who has defined and solved this problem was Grunert in 1841.

Let the unknown positions of the three points of the known triangle be

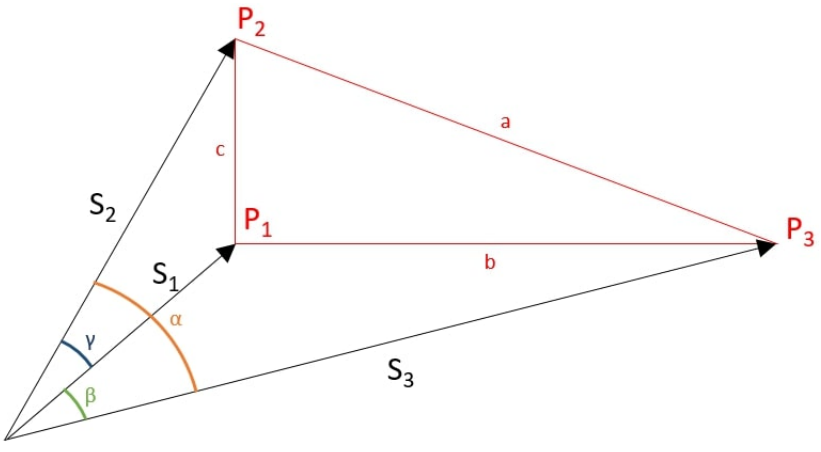
$$P_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \qquad i = 1, 2, 3$$

Let the known side lengths of the triangle be

$$a = \|P_2 - P_3\|$$

$$b = \|P_1 - P_3\|$$

$$c = \|P_1 - P_2\|$$



The center of perspectivity would be the origin of the camera reference frame, while the image projection plane would be a distance f in front of the center of perspectivity.

The observed perspective projection of P_i are

$$q_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix} \qquad i = 1, 2, 3 \qquad u_i = f\frac{x_i}{z_i} \qquad v_i = f\frac{y_i}{z_i}$$

The unit vectors j_i pointing from the center of perspectivity to the observed points P_i are

$$j_i = \frac{1}{\sqrt{u_i^2 + v_i^2 + f^2}} \begin{bmatrix} u_i \\ v_i \\ f \end{bmatrix} \qquad i = 1, 2, 3$$

Moreover, $\cos\alpha = j_2 \cdot j_3$, $\cos\beta = j_1 \cdot j_3$, $\cos\gamma = j_1 \cdot j_2$. Eventually, let the unknown distances of points P_i from the center of perspectivity be $s_i = \|P_i\|$, where again $i = 1, 2, 3$.
In order to determine the position of P_1, P_2 and P_3 with respect to the camera reference frame, it's sufficient to determine s_1, s_2 and s_3 because $P_i = s_i j_i$ for $i = 1, 2, 3$.

4.4.2 Solutions

As already mentioned, the first one who provided a solution was Grunert in 1841. His starting point was the law of cosines:

$$s_2^2 + s_3^2 \cos\alpha = a^2$$

$$s_1^2 + s_3^2 \cos\beta = b^2$$

$$s_1^2 + s_2^2 \cos\gamma = c^2$$

Knowing that $s_2 = u s_1$ and $s_3 = v s_1$, he concluded that $u = \dfrac{\left(-1 + \frac{a^2-c^2}{b^2}\right) v^2 - 2\left(\frac{a^2-c^2}{b^2}\right)\cos\beta v + 1 + \frac{a^2-c^2}{b^2}}{2(\cos\gamma - v\cos\alpha)}$
From this expression of u, a fourth order polynomial in v can be obtained, which can have as many as four real roots:

$$A_4 v^4 + A_3 v^3 + A_2 v^2 + A_1 v + A_0 = 0$$

Thanks to the expression of u previously reported, to every solution for v there will be a corresponding solution for u. Then, knowing u and v, it's easy to determine s_1, s_2 and s_3.
In 1903, Finsterwalder provides a solution in which the root weren't found in a fourth order polynomial, but rather he found a root from a cubic polynomial and the roots of two quadratic polynomials:

$$Au^2 + 2Buv + Cv^2 + 2Du + 2Ev + F = 0$$

In 1949, unaware of the Grunert and the Finsterwalder solutions, Merritt obtained a fourth order polynomial as well:

$$a^2 s_1^2 - b^2 s_2^2 + (a^2 - b^2) s_3^2 - 2a^2 s_1 s_3 \cos\beta + 2b^2 s_2 s_3 \cos\alpha = 0$$

and

$$a^2 s_1^2 + (a^2 - c^2) s_2^2 - c^2 s_3^2 - 2a^2 s_1 s_2 \cos\gamma + 2c^2 s_2 s_3 \cos\alpha = 0$$

Then, he obtained the following equations:

$$-b^2u^2 + (a^2 - b^2)v^2 - 2a^2\cos\beta v + 2b^2\cos\alpha uv + a^2 = 0$$

and

$$(a^2 - c^2)u^2 - c^2v^2 - 2a^2\cos\gamma u + 2c^2\cos\alpha uv + a^2 = 0$$

In this last equation he substituted the value $v = \dfrac{b^2 - a^2 - c^2 + (b^2 + c^2 - a^2)u^2 + 2(a^2 - b^2)\cos\gamma u}{2c^2(u\cos\alpha - \cos\beta)}$, producing the fourth order polynomial equation

$$B_4u^4 + B_3u^3 + B_2u^2 + B_1u + B_0 = 0$$

In 1981, Fischler and Bolles obtained these two equation, which are very similar to those of Grunert, but in a different form:

$$\left(1 - \frac{a^2}{b^2}\right)v^2 + 2\left(\frac{a^2}{b^2}\cos\beta - \cos\alpha u\right)v + u^2 - \frac{a^2}{b^2} = 0$$

and

$$v^2 - 2\cos\alpha uv + \left(1 - \frac{a^2}{c^2}\right)u^2 + 2\frac{a^2}{c^2}\cos\gamma u - \frac{a^2}{c^2} = 0$$

He also achieved a fourth order polynomial equation

$$D_4u^4 + D_3u^3 + D_2u^2 + D_1u + D_0 = 0$$

In 1989, a team composed by Grafarend, Lohse and Schaffrim started their work from the following two equations (they were aware of all the previous work, except for the Fischler - Bolles solution):

$$-\frac{a^2}{c^2}s_1^2 + \left(1 - \frac{a^2}{c^2}\right)s_2^2 + s_3^2 + 2\frac{a^2}{c^2}s_1s_2\cos\gamma - 2\cos\alpha s_2s_3 = 0$$

and

$$\left(1 - \frac{b^2}{c^2}\right)s_1^2 - \frac{b^2}{c^2}s_2^2 + s_3^2 + 2\frac{b^2}{c^2}\cos\gamma s_1s_2 - 2\cos\beta s_1s_3 = 0$$

The year before, in 1988, Linnainmaa, Harwood and Davis provided another direct solution. Making a change of variable in the first equation of Grunert, they obtained

$$(1 - \cos^2\beta)s_1^2 + v^2 = b^2$$

$$(1 - \cos^2\gamma)s_1^2 + u^2 = c^2$$

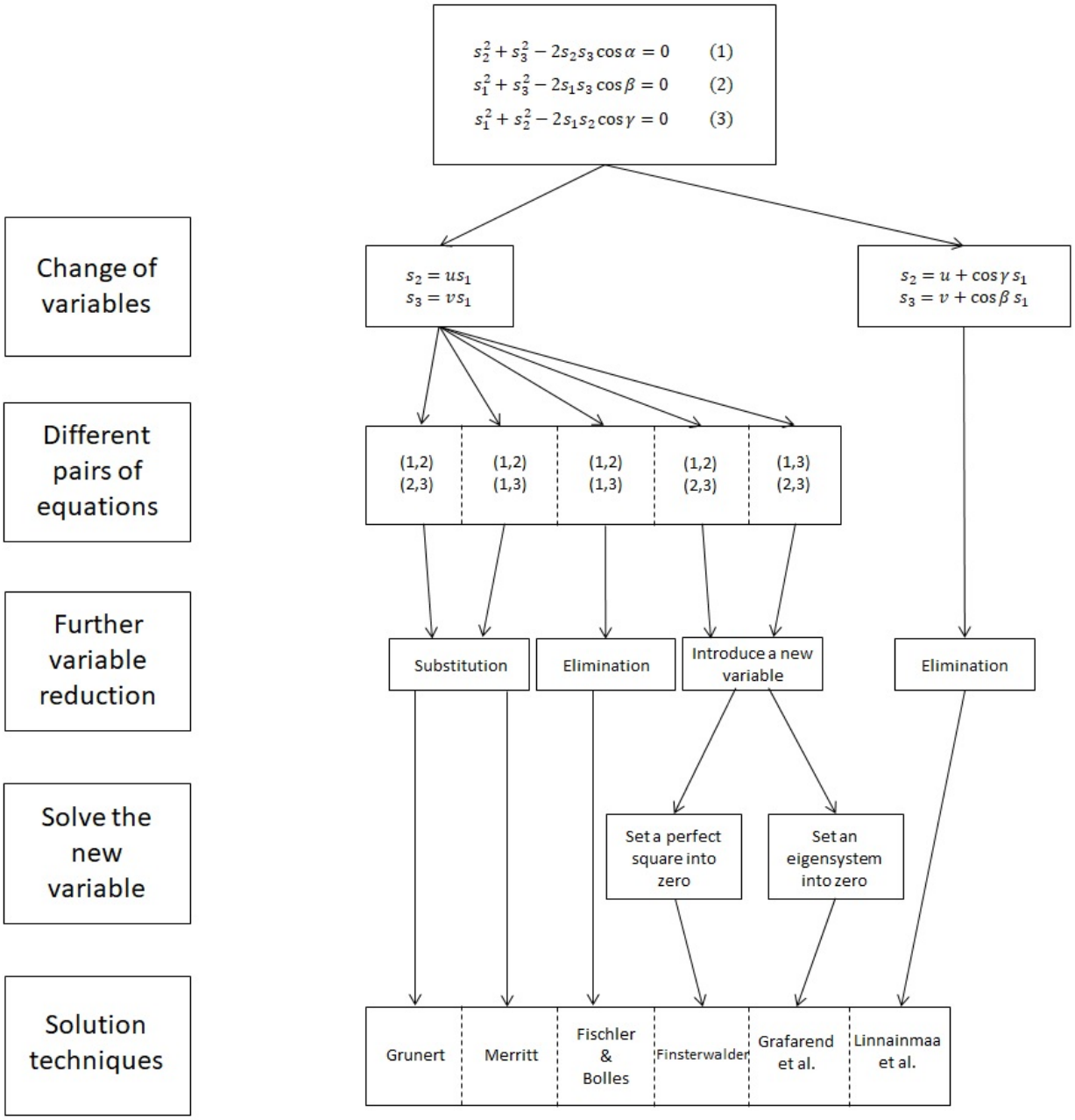
By making the right substitution and simplification, the polynomial achieved was

$$r_1s_1^4 + r_2s_1^2 + r_3 = (r_4s_1^2 + r_5)uv$$

To eliminate the uv term, they squared and simplified this equation, obtaining a polynomial like

$$t_8 s_1^8 + t_6 s_1^6 + t_4 s_1^4 + t_2 s_1^2 + t_0 = 0$$

Since s_1 must be positive, there are at most four solutions to this equation. Once a value for s_1 has been determined, they solved in u and v the following equations: $q_1 s_1^2 + u^2 = c^2$ and $q_2 s_1^2 + v^2 = b^2$. Eventually, in order to find the positive solutions for s_2 and s_3, they substituted the just found u and v in $s_2 = u + \cos\gamma s_1$ and $s_3 = v + \cos\beta s_1$.


$s_2^2 + s_3^2 - 2s_2s_3\cos\alpha = 0$ (1)
$s_1^2 + s_3^2 - 2s_1s_3\cos\beta = 0$ (2)
$s_1^2 + s_2^2 - 2s_1s_2\cos\gamma = 0$ (3)
Change of variables
$s_2 = us_1$
$s_3 = vs_1$
$s_2 = u + \cos\gamma\, s_1$
$s_3 = v + \cos\beta\, s_1$
Different pairs of equations
(1,2) (2,3)
(1,2) (1,3)
(1,2) (1,3)
(1,2) (2,3)
(1,3) (2,3)
Further variable reduction
Substitution
Elimination
Introduce a new variable
Elimination
Solve the new variable
Set a perfect square into zero
Set an eigensystem into zero
Solution techniques
Grunert
Merritt
Fischler & Bolles
Finsterwalder
Grafarend et al.
Linnainmaa et al.

4.4.3 Comparisons of the solutions

Different pairs of equations are used by Grunert and Merrit: therefore, the coefficients in their fourth order polynomials are different. However, from the algebraic point of view, their solutions are identical. Fischler and Bolles just multiply some terms to two pairs of equations and then subtract each other without expressing one variable in terms of the other, while Grunert and Merritt use the substitution to reduce the two variables into one. The substitution is the easiest method for reducing the number of variables. Conversely, a tougher method is the direct elimination: Fischler and Bolles and Linnainmaa et al. proceed in this way. Linnainmaa et al. use $s_2 = u + \cos\gamma s_1$ and $s_3 = v + \cos\beta s_1$ as the change of variables; Finsterwalder and Grafarend et al. introduce the same variable, but they use different approaches to solve λ. Finsterwalder solves the equation $Au^2 + 2Buv + Cv^2 + 2Du + 2Ev + F = 0$ for v and seeks a λ to make the term inside the square root be a perfect square. On the contrary, Grafarend et al. try to solve the eigensystem $(s_1 s_2 s_3)(P - \lambda Q)(s_1 s_2 s_3) = 0$. At this point these two approaches are algebraically equivalent.

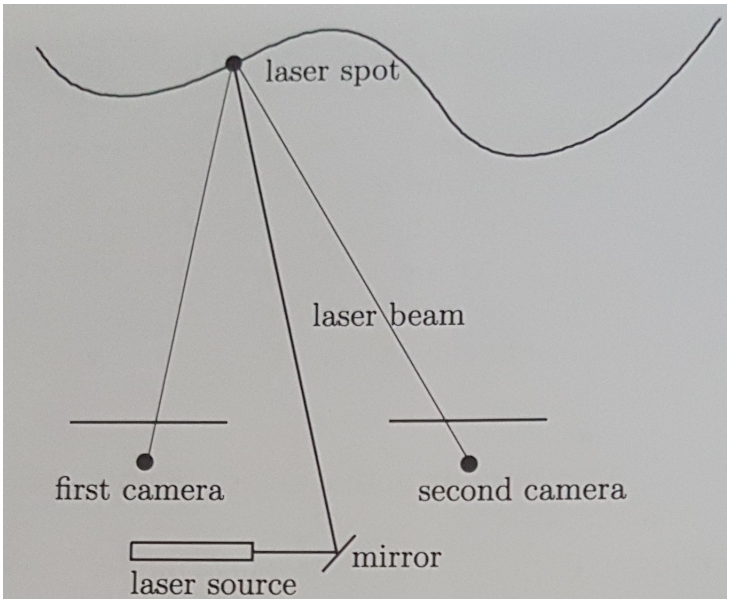
In any case, there exist some geometric structures for the three point space resection on which the resection is unstable or indeterminate. Clearly, if the structure is unstable, the position of the three vertices will change a lot even if the position of the center of perspectivity undergoes a small variation. Besides the singularity caused by geometric structures, there also exist some singularities caused by the algebraic derivation of solutions.

Authors	Features	Algebraic Singularities
Grunert 1841	Direct solution, solve a fourth order polynomial	Yes
Finsterwalder 1903	Form a cubic polynomial and find the roots of two quadratics	Yes
Merritt 1949	Direct solution, solve a fourth order polynomial	Yes
Fischler and Bolles 1981	Another approach to form a fourth order polynomial	No
Linnainmaa et al. 1988	Generate an eighth order polynomial	No
Grafarend et al. 1989	Form a cubic polynomial and find intersection of two quadratics	Yes

4.5 Engineering application by Faugeras: pose of 3D objects

In this section we will concentrate on the use of laser rangefinder, while in the next we will use the stereo [18].

A laser source produces a beam that falls on a system of moving mirrors that is used to control its direction in space. A bright spot on the rigid bodies is produced by the laser beam and it's imaged by a number of cameras forming a calibrated stereo rig. Then, the 3D position of the spot is reconstructed by triangulation: this is the principle of active stereo.

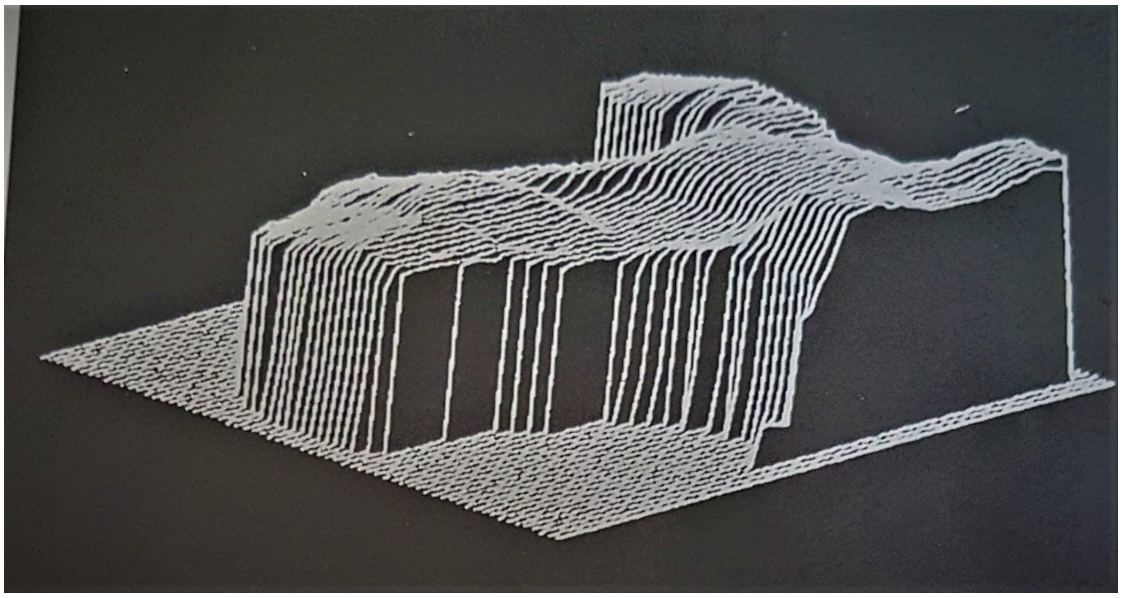


The rigid bodies are placed on a computer-controlled table with 2 DOFs: a rotation and a translation with respect to the same vertical axis. Another different system with two moving mirrors with their axes at right angles has been buit as well. In this case, three cameras are used, because there is an extra DOF.

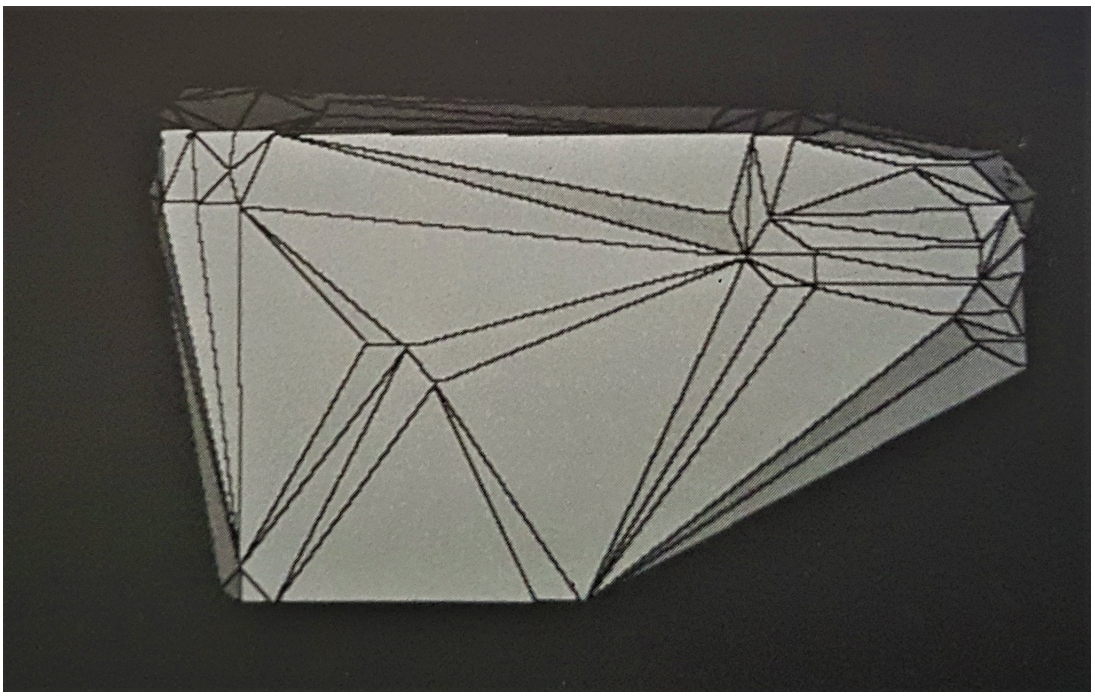
In any case, each system produces a set of 3D points M represented by their coordinates x, y and z in a reference frame attached to the stereo rig.

As a furhter example, let us consider the following photograph of a Renault part.

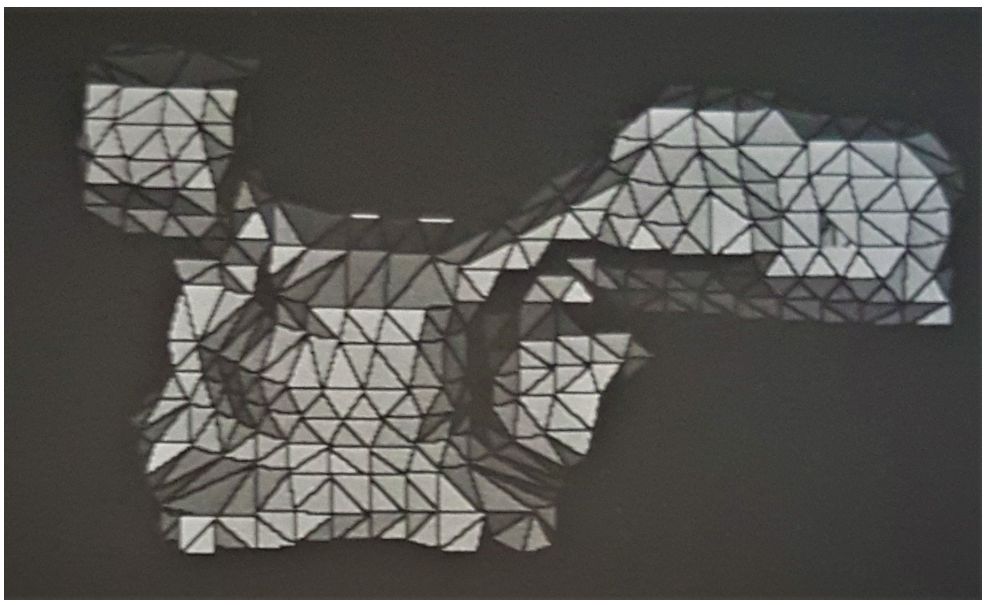
Then, consider one set of measured 3D points on its surface:

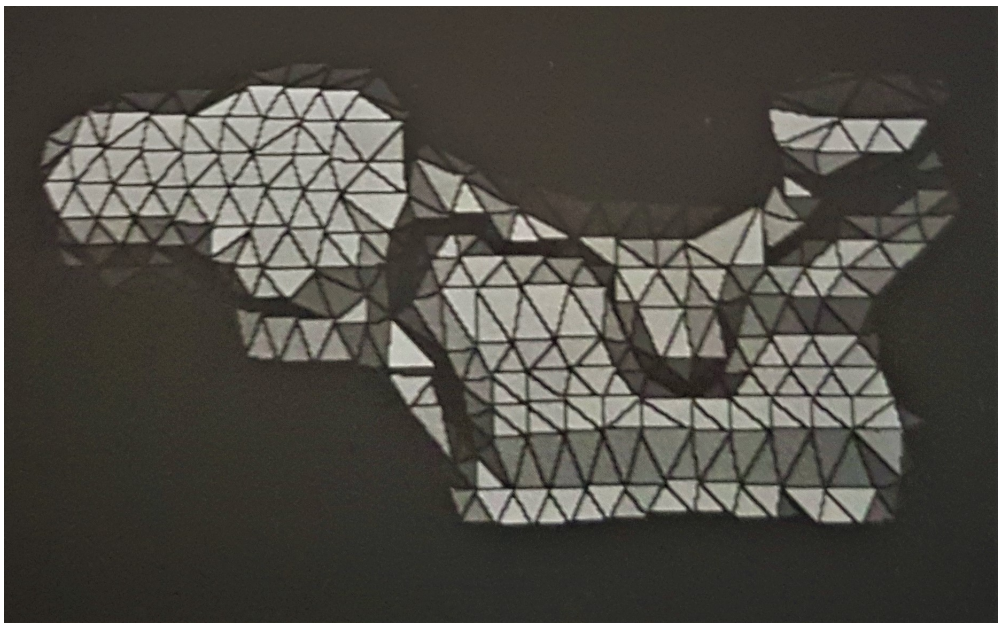


and eventually the boundary of the original Delaunay triangulation (wich is the convex hull of the measured points):



If now we proceed with the triangulation of the object, we will obtain the following two images from two different points of view:





The transformation T, which allow us to pass from the reference systems of the models to those of the scenes, is represented by a 6D vector $\mathbf{a} = [\mathbf{r}', \mathbf{t}']'$, where $\mathbf{r}$ and $\mathbf{t}$ are the vectors representing the rotation and the translation.
The plane is represented by a 3D vector $\mathbf{p} = [a, b, c]'$ in one of three possible maps. [**?**]

4.5.1 The effect of T applied on p

Consider the matrix

$$\mathbf{D} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \qquad \mathbf{D}^{-1} = \begin{bmatrix} \mathbf{R}' & \mathbf{-R't} \\ \mathbf{0} & 1 \end{bmatrix}$$

Let us suppose that the plane is represented in the map ϕ_1. The coefficients of the equation of the transformed plane are

$$(\mathbf{D}^{-1})' = \begin{bmatrix} 1 \\ \mathbf{p} \end{bmatrix}$$

If we want to represent the transformed plane in ϕ_1, we can write

$$a' = \frac{\mathbf{r}_2'\mathbf{m}}{\mathbf{r}_1'\mathbf{m}} \qquad b' = \frac{\mathbf{r}_3'\mathbf{m}}{\mathbf{r}_1'\mathbf{m}} \qquad c' = \frac{-\mathbf{t}'\mathbf{R}\mathbf{m} + c}{\mathbf{r}_1'\mathbf{m}}$$

where $\mathbf{r}_i', i = 1, 2, 3$ are the row vectors of $\mathbf{R}$, $\mathbf{m} = [1, a, b]'$ is a vector normal to the plane and $\mathbf{p}_1 = [a', b', c']'$ is the transformed plane.
These three equations define a measurement equation $\mathbf{f}(\mathbf{x}, \mathbf{a}) = \mathbf{0}$, which is used when the model plane represented by $\mathbf{p}$ is matched to the scene plane repesented by $\mathbf{p}_1$, where $\mathbf{x} = [\mathbf{p}', \mathbf{p}_1']'$ and $\mathbf{a}$ is the above representation of T.

4.5.2 Finding the initial rotations

Let M_1 and M_2 be the model planes; let S_1 and S_2 be the scene planes. Let $\mathbf{n}_1$, $\mathbf{n}_2$, $\mathbf{n}'_1$ and $\mathbf{n}'_2$ be the corresponding unit normals.

We need to find a rotation matrix $\mathbf{R}$ such that

$$C = \sum_{i=1}^{2} \|\varepsilon'_i \mathbf{n}'_i - \varepsilon_i \mathbf{R}\mathbf{n}_i\|^2 = 0$$

where $\varepsilon_i, \varepsilon'_i = \pm 1$.

Assuming the more general case and denoting by $\mathbf{q}$ one of the two quaternions representing $\mathbf{R}$, we obtain

$$C = \sum_{i=1}^{2} |\mathbf{n}'_i - \mathbf{q} \times \mathbf{n}_i \times \bar{\mathbf{q}}|^2$$

Multiplying the previous equation by $|\mathbf{q}|^2$, and doing some adjustments, we obtain

$$C = \sum_{i=1}^{2} |\mathbf{q}' \mathbf{A}'_i \mathbf{A}_i \mathbf{q}|$$

where $\mathbf{A}_i$ is a 4x4 matrix. Therefore, the problem can be written as $min_q \mathbf{q}'\mathbf{A}\mathbf{q}$, where $\mathbf{A} = \sum_{i=1}^{2} \mathbf{A}'_i \mathbf{A}_i$. We don't demonstrate that the solution to this problem is the eigenvector of unit length of matrix $\mathbf{A}$ corresponding to the smallest eigenvalue.

4.5.3 Finishing the search

We build a number of recognition sequences $R_n^i = ((M_1, S_{i_1}), ..., (M_n, S_{i_n}))$, each one characterized by a transformation T^i represented by $\mathbf{a}^i$ and weighted by $\mathbf{P}^i$.

$$\mathbf{d}_j = \mathbf{f}(\mathbf{x}_j, \mathbf{a}^i) \Lambda_j^{-1} \mathbf{f}(\mathbf{x}_j, \mathbf{a}^i)' \qquad \Lambda_j = \frac{\partial \mathbf{f}}{\partial \mathbf{a}} \mathbf{p}^i \frac{\partial \mathbf{f}}{\partial \mathbf{a}'} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{U}_j \frac{\partial \mathbf{f}}{\partial \mathbf{x}'}$$

The error corresponding to the recognition sequence R^i is then

$$\varepsilon_n^i = \frac{1}{n} \sum_{j=1}^{n} d_j$$

Conclusions

Through this work, I explained the main characteristics of rotations, describing them by matrix and quaternion representations. Then, I reported some applications of Euler angles, special unitary matrices and quaternions. Moreover, I described the algebra of both Lie groups and quaternions. Eventually, I discussed the problem of pose estimation and I reported some of the most important researches on this field since XIX century, especially focusing on the one provided by Faugeras.

Bibliography

[1] G.Parigi, A.Palestini, *Manuale di geometria*, Pitagora Editrice, Bologna (2007).

[2] S.L. Altmann, *Rotations, quaternions, and double groups*, Courier Corporation (2005).

[3] http://www.euclideanspace.com/maths/geometry/rotations/euler/

[4] J. Diebel, *Representing attitude: Euler angles, unit quaternions, and rotation vectors*, Matrix 15-16 (2006), 1–35.

[5] G.G. Slabaugh, *Computing Euler angles from a rotation matrix*, Retrieved on August 2000 (1999), 39–63.

[6] K. Shoemake, *Animating rotation with quaternion curves*, ACM SIGGRAPH computer graphics 3 (1985), 245–254.

[7] W. Schmid, *Lie groups and Lie algebras* (2012).

[8] N. Bourbaki, *Lie groups and Lie algebras*, Springer Science & Business Media (2008).

[9] D. Bump, *Lie groups*, Springer (2004).

[10] A. Kirillov, *Introduction to Lie Groups and Lie Algebras*.

[11] C. A. Deavours, *The quaternion calculus*, The American Mathematical Monthly 9 (1973), 995–1008.

[12] E. Trucco, A. Verri, *Introductory techniques for 3-D computer vision*, Prentice Hall Englewood Cliffs (1998).

[13] https://www.quantamagazine.org/the-strange-numbers-that-birthed-modern-algebra-20180906/

[14] J. Kuipers, *Quaternions and rotation sequences*, Princeton university press Princeton (1999).

[15] M. Arribas, A. Elipe, M. Palacios, *Quaternions and the rotation of a rigid body*, Celestial Mechanics and Dynamical Astronomy 3-4 (2006), 239–251.

[16] R. M. Haralick, H. Joo, C. Lee, X. Zhuang, V. G. Vaidya, M. B. Kim, *Pose estimation from corresponding point data*, IEEE Transactions on Systems, Man, and Cybernetics 6 (1989), 1426–1446.

[17] B. M. Haralick, K. Ottenberg, C. Lee, M. Nölle, *Review and analysis of solutions of the three point perspective pose estimation problem*, International journal of computer vision 3 (1994), 331–356.

[18] O. Faugeras, *Three-dimensional computer vision: a geometric viewpoint*, MIT press (1993).