6.801/6.866: Machine Vision, Lecture 18

Professor Berthold Horn, Ryan Sander, Tadayuki Yoshitake
MIT Department of Electrical Engineering and Computer Science
Fall 2020

These lecture summaries are designed to be a review of the lecture. Though I do my best to include all main topics from the lecture, the lectures will have more elaborated explanations than these notes. Therefore, if you're looking for the most rigorous review and treatment of these topics, we encourage you to rewatch the lecture videos. With that said, we hope these summaries are beneficial for your learning. If you have any feedback for these lecture summaries, please submit it **here**.

1 Lecture 18: Rotation and How to Represent it, Unit Quaternions, the Space of Rotations

Today, we will focus on rotations. Note that unlike **translations**, **rotations** are not **commutative**, which makes fitting estimates to data, amongst other machine vision tasks, more challenging. In this lecture, we will cover rotation in terms of:

- Properties
- Representations
- Hamilton's Quaternions
- Rotation as Unit Quaternion
- Space of rotations
- Photogrammetry
- Closed-form solution of absolute quaternions
- Divison algebras, quaternion analysis, space-time

We will start by looking at some motivations for why we might care about how we formulate and formalize rotations: **What is rotation used for?**

- Machine vision
- Recognition/orientation
- Graphics/CAD
- Virtual Reality
- Protein Folding
- Vehicle Attitude
- Robotics
- Spatial Reasoning
- Path Planning - Collision Avoidance

1.1 Euclidean Motion and Rotation

Rotation and translation form Euclidean motion. Some properties of Euclidean motion, including this property. Euclidean motion:

- Contains **translation** and **rotation**
- Preserves distances between points
- Preserves angles between lines
- Preserves handedness
- Preserves dot-products
- Preserves triple products
- Does not contain:
  - Reflections
  - Skewing
  - Scaling

1.2 Basic Properties of Rotation

Now that we have framed rotation to be a property of Euclidean motion, we can dive further into some properties of rotation:

- **Euler's Theorem**: There is a line of fixed points that remain the same in any rotation. This is known as the axis of rotation.
- **Parallel axis theorem**: Any rotation is equivalent to rotation through the origin along with a translation. This allows for the decoupling between translation and rotation.
- **Rotation of sphere includes rotation of space**: I.e, any rotation that is induced on a space can be induced on an equivalent-dimensional sphere.
- **Attitude, Orientation - Rotation Relative to Reference**: This helps for determining the orientation of objects relative to another frame of reference.
- **Degrees of Freedom (in 3D)** is 3.

Some additional properties that are helpful for understanding rotation:

- **Rotational velocity** can be described by a vector. This set of rotational vectors together form a **lie algebra**, specifically, $\mathbf{so}(3)$. The set of rotations form the counterpart to this lie algebra, namely the **Special Orthogonal group** $\mathbf{SO}(3)$, which is a **Lie group**. If you are not familiar with Lie groups/algebras, here are a few useful terms to get started with these concepts:
  - A **group** is a set equipped with a binary operation that combines any two elements to form a third element in such a way that four conditions called group axioms are satisfied, namely closure, associativity, identity and invertibility [1].
  - A **manifold** is a d-dimensional topological space that locally resembles Euclidean space at each point i.e. any patch of the manifold can be "locally" approximated as a d-dimensional Euclidean space [2].
  - A **Lie group** is a group that is also a differentiable manifold. One key Lie group that we will study in this course is the **Special Orthogonal group**, known as $\mathbf{SO}(3)$, which is the space of orthonormal rotation matrices.
- We can define the derivative of the radius vector $\dot{\mathbf{r}}$ using **Poisson's Formula**: $\dot{\mathbf{r}} = \hat{\boldsymbol{\omega}} \times \mathbf{r}$
- **Rotational velocities add** - this holds because these vectors belong to **Lie algebras**
- **Finite rotations do not commute** - this holds because rotations are defined by **Lie groups** in a non-Euclidean space.
- The **Degrees of Freedom** for rotation in dimension $\mathbf{n}$ is given by $\frac{n(n-1)}{2}$, which coincidentally equals 3 in 3 dimensions.
- **Intuition**: Oftentimes, it is easier to think about rotation "in planes", rather than "about axes". Rotations preserve points in certain planes.

1.2.1 Isomorphism Vectors and Skew-Symmetric Matrices

A technical note that is relevant when discussing cross products: Although a cross product produces a vector in 3D, in higher dimensions the result of a cross product is a **subspace**, rather than a vector.

Specifically, this **subspace** that forms the result of a higher-dimensional cross product is the space that is **perpendicular/orthogonal** to the two vectors the cross product operator is applied between.

With this set up, we can think of cross products as producing the following isomorphism vectors and skew-symmetric matrices: **One Representation**:

$$\mathbf{a} \times \mathbf{b} = \mathbf{A}\mathbf{b}$$
$$\mathbf{A} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

An Isomorphic Representation:

$$\mathbf{a} \times \mathbf{b} = \bar{\mathbf{B}}\mathbf{a}$$
$$\mathbf{A} = \begin{bmatrix} 0 & b_z & -b_y \\ -b_z & 0 & b_x \\ b_y & -b_x & 0 \end{bmatrix}$$

Note that while these skew-symmetric matrices have 9 elements as they are 3×3 matrices, they only have 3 DOF.

1.3 Representations for Rotation

There are a myriad of representations for rotations, highlighting that different applications/domains/problem-solving techniques demand different representations for these transformations. Some of these representations include (we will proceed in greater detail about these later):

1. Axis and angle
2. Euler Angles
3. Orthonormal Matrices
4. Exponential cross product
5. Stereography plus bilinear complex map
6. Pauli Spin Matrices
7. Euler Parameters
8. Unit Quaternions

Let us delve into each of these in a little more depth.

1.3.1 Axis and Angle

This representation is composed of a vector $\hat{\boldsymbol{\omega}}$ and an angle θ, along with the **Gibb's vector** that combines these given by $\hat{\boldsymbol{\omega}} \tan\left(\frac{\theta}{2}\right)$, which has magnitude $\tan\left(\frac{\theta}{2}\right)$, providing the system with an additional degree of freedom that is not afforded by unit vectors. Therefore, we have our full 3 rotational DOF.

1.3.2 Euler Angles

A few notes about these:

- There are over 24 definitions of Euler angles! This is the case because these definitions are permutations - i.e. the order of the angles matters here. The order of angular composition matters.
- These rotations are defined through each axis, **roll**, **pitch**, and **yaw**.

1.3.3 Orthonormal Matrices

We have studied these previously, but these are the matrices that have the following properties:

1. $\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = I, \mathbf{R}^T = \mathbf{R}^{-1}$ (skew-symmetric)
2. $\det |R| = +1$
3. $\mathbf{R} \in \mathbf{SO}(3)$ (see notes on groups above) - being a member of this Special Orthogonal group is contingent on satisfying the properties above.

1.3.4 Exponential Cross Product

We can also write rotations in terms of a matrix exponential. To derive this matrix exponential, let us consider the matrix first-order, homogeneous differential equation defined by:

$$\frac{d\mathbf{R}}{d\theta} = \mathbf{\Omega}\mathbf{R}$$

Has the solution given by the matrix exponential:

$$\mathbf{R} = e^{\theta\mathbf{\Omega}}$$

Taking the Taylor expansion of this expression, we can write this matrix exponential as:

$$\begin{aligned}\mathbf{R} &= e^{\theta\mathbf{\Omega}} \\ &= \sum_{i=0}^{\infty}\frac{1}{i!}(\theta\mathbf{\Omega})^i \\ &= \sum_{i=0}^{\infty}\frac{\theta^i}{i!}(\mathbf{V}_\Omega\mathbf{\Lambda}_\Omega^i\mathbf{V}_\Omega^T) \;\; \textbf{(Taking an eigendecomposition of } \mathbf{\Omega} = \mathbf{V}_\Omega\mathbf{\Lambda}_\Omega\mathbf{V}_\Omega^T\textbf{)}\end{aligned}$$

(Optional) Let us look a little deeper into the mathematics behind this exponential cross product. We can write a rotation about $\hat{\boldsymbol{\omega}}$ through angle θ as:

$$\begin{aligned}&\mathbf{r} = R(\theta)\mathbf{r}_0 \\ &\frac{d\mathbf{r}}{d\theta} = \frac{d}{d\theta}(R(\theta)\mathbf{r}_0) \\ &\frac{d\mathbf{r}}{d\theta} = \hat{\boldsymbol{\omega}} \times \mathbf{r} = \mathbf{\Omega}\mathbf{r} = \mathbf{\Omega}R(\theta)\mathbf{r}_0 \\ &\frac{d}{d\theta}R(\theta)\mathbf{r}_0 = \mathbf{\Omega}R(\theta)\mathbf{r}_0\end{aligned}$$

Then for all $\mathbf{r}_0$:

$$\frac{d}{d\theta}R(\theta) = \mathbf{\Omega}R(\theta) \implies R(\theta) = e^{\theta\mathbf{\Omega}}$$

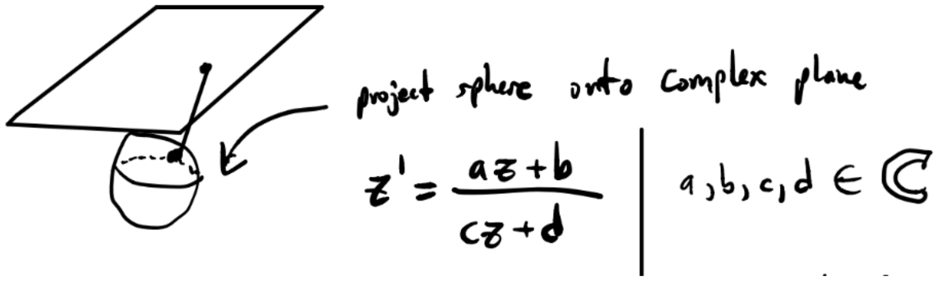
1.3.5 Stereography Plus Bilinear Complex Map

Intuitively, this notion of rotation corresponds to mapping a sphere to a complex plane, making transformations in the complex plane, and then mapping this transformed mapping from the complex plane back to the sphere. A few points here:

- This is an instance of **Stereography**: a **conformal** (angle-preserving) spherical mapping.
- The rotation of a sphere induces the rotation of a space.
- This plane is treated as a complex plane that we can apply homogeneous transforms to.

We can conceptually visualize this mapping from a sphere to the complex plane using the figure below:

Figure 1: Mapping from a sphere to a complex plane, which we then apply a homogeneous transformation to and map back to the sphere in order to induce a rotation.


project sphere onto complex plane
$z' = \frac{az+b}{cz+d}$
$a, b, c, d \in \mathbb{C}$

Specifically, the homogeneous transform we consider maps the complex variable z to z':

$$z' = \frac{az+b}{cz+d}, \text{ for some } a, b, c, d \in \mathbb{C}$$

We can actually generalize this framework even further - any rotation in 3-space can be thought of as an operation in a complex plane.

1.3.6 Pauli Spin Matrices

With a physical motivation, these are 2×2 complex-valued, unitary, Hermitian matrices ($\mathbf{A} = \bar{\mathbf{A}}^T$). All these constraints create 3 DOF for these matrices:

$$\mathbf{S}_x = \frac{\hbar}{2}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \mathbf{S}_y = \frac{\hbar}{2}\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \mathbf{S}_z = \frac{\hbar}{2}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

1.3.7 Euler Parameters

There are also known as **Rodrigues** parameters, leading to **Rodrigues formula**, which can be thought of as a rotation about the unit vector $\hat{\boldsymbol{\omega}}$ through the angle θ:

$$\mathbf{r}' = (\cos\theta)\mathbf{r} + (1-\cos\theta)(\hat{\boldsymbol{\omega}} \cdot \mathbf{r})\hat{\boldsymbol{\omega}} + \sin\theta(\hat{\boldsymbol{\omega}} \times \mathbf{r})$$

The geometry of this problem can be understood through the following figure:

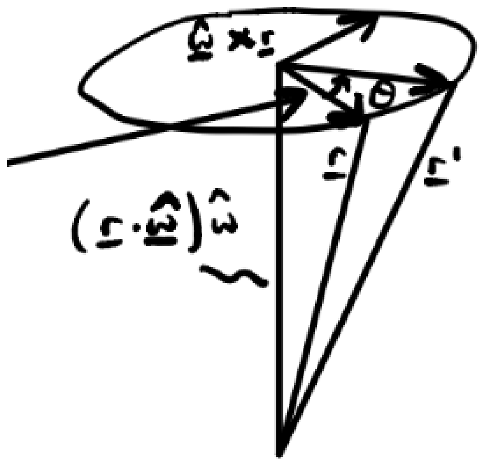


Figure 2: Geometric interpretation of the Rodrigues formula: Rotation about the vector $\hat{\boldsymbol{\omega}}$ through an angle θ.

One disadvantage of this approach is that there is no way to have **compositions** of rotations.

Next, let us take an in-depth analysis of the Rodrigues Formula and the Exponential Cross Product:

$$\frac{d\mathbf{R}}{d\theta} = \mathbf{\Omega R} \implies \mathbf{R} = e^{\mathbf{\Omega}\theta}$$

$$e^{\mathbf{\Omega}\theta} = \mathbf{I} + \theta\mathbf{\Omega} + \frac{1}{2!}(\theta\mathbf{\Omega})^2 + ... = \sum_{i=0}^{\infty}\frac{\theta^i}{i!}\mathbf{\Omega}^i$$

Next, we have that:

$$\begin{aligned}\mathbf{\Omega}^2 &= (\hat{\boldsymbol{\omega}}\hat{\boldsymbol{\omega}}^T - \mathbf{I}) \\ \mathbf{\Omega}^3 &= -\mathbf{\Omega}\end{aligned}$$

We can then write this matrix exponential as:

$$\begin{aligned}e^{\theta\mathbf{\Omega}} &= \mathbf{I} + \mathbf{\Omega}(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} + \cdots) + \mathbf{\Omega}^2(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} + \cdots) \\ &= \mathbf{I} + \mathbf{\Omega}(\sum_{i=0}^{\infty} \frac{\theta^{2i+1}}{(2i+1)!}(-1)^i) + \mathbf{\Omega}^2(\sum_{i=0}^{\infty} \frac{\theta^{2i+2}}{(2i+2)!}(-1)^i) \\ &= \mathbf{I} + \mathbf{\Omega}\sin\theta + \mathbf{\Omega}^2(1-\cos\theta) \\ &= \mathbf{I} + (\sin\theta)\mathbf{\Omega} + (\hat{\boldsymbol{\omega}}\hat{\boldsymbol{\omega}}^T - \mathbf{I})(1-\cos\theta) \\ &= (\cos\theta)\mathbf{I} + (\sin\theta)\mathbf{\Omega} + (1-\cos\theta)\hat{\boldsymbol{\omega}}\hat{\boldsymbol{\omega}}^T\end{aligned}$$

From this, we have:

$$\begin{aligned}\mathbf{r} &= e^{\theta\mathbf{\Omega}}\mathbf{r} \\ \mathbf{r}' &= (\cos\theta)\mathbf{r} + (1-\cos\theta)(\hat{\boldsymbol{\omega}}\cdot\mathbf{r})\hat{\boldsymbol{\omega}} + \sin\theta(\hat{\boldsymbol{\omega}}\times\mathbf{r})\end{aligned}$$

Where the last line is the result of the Rodrigues formula.

1.4 Desirable Properties of Rotations

We would like rotations to exhibit the following properties:

- The ability to rotate vectors - or coordinate systems
- The ability to compose rotations
- Intuitive, non-redundant representation - e.g. rotation matrices have 9 entries but only 3 degrees of freedom
- Computational efficiency
- Interpolate orientations
- Averages over range of rotations
- Derivative with respect to rotations - e.g. for optimization and least squares
- Sampling of rotations - uniform and random (e.g. if we do not have access to closed-form solutions)
- Notion of a space of rotations

1.5 Problems with Some Rotation Representations

Let us discuss some problems with the representations we introduced and discussed above:

1. **Orthonormal Matrices**: Redundant, with complex constraints.
2. **Euler Angles**: Inability to compose rotations, "gimbal lock" (when two axes line up, you lose a DOF, which can be disastrous for control systems).
3. **Gibb's Vector**: Singularity when $\theta = \pi$, since $\tan(\frac{\pi}{2})$ is not defined.
4. **Axis and Angle**: Inability to compose rotations
5. There is no notion of the "space of rotations" (at least not in Euclidean space).

How can we overcome these challenges? One way through which we can do is to look at another representation approach, this time through Hamilton.

1.6 Quaternions

In this section, we will discuss another way to represent rotations: quaternions.

1.6.1 Hamilton and Division Algebras

Goal: Develop an algebra where we can use algebraic operations such as addition, subtraction, mulitplication, and division.

1. Hamilton's representation is motivated by **Algebraic couples**, for instance, complex numbers as pairs of reals $a+bi, a-bi$
2. **Here, we want:** Multiplicative inverses
3. **Examples**: Real numbers, complex numbers
4. **Expected next**: Three conponents (vectors) - note that this was before Gibb's vector and 3-vectors

Hamilton's insight here is that these quaternions require a "4th dimension for the sole purpose of calculating triples".

1.6.2 Hamilton's Quaternions

Hamilton noted the following insights in order to formalize his quaternion:

- Quaternions cannot be constructed with three components
- Quaternions need additional square roots of -1

Thereofore, the complex components i, j, k defined have the following properties:

1. $i^2 = j^2 = k^2 = ijk = -1$
2. From which follows:
   (a) $ij = k$
   (b) $jk = i$
   (c) $ki = j$
   (d) $ji = -k$
   (e) $kj = -i$
   (f) $ik = -j$

   Note: As you can see from these properties, multiplication of the components of these quaternions is not commutatitve.

1.6.3 Representations of Quaternions

There are several ways through which we can represent these quaternions:

1. **Real and imaginary parts**: $q_0 + iq_x + jq_y + kq_z$
2. **Scalar and 3-vector**: $(q, \mathbf{q})$
3. **4-vector**: $\overset{o}{q}$
4. **Certain Orthogonal 4 x 4 Matrices q** (this is an isomorphism of quaternions that allows us to do multiplications).
5. **Complex Composite of Two Complex Numbers**: Here, we have $(a + bi)$ and $(c + di)$, and we replace all the real constants a, b, c, d with complex numbers. We can build sophisticated algebras from these operations.

1.6.4 Representations for Quaternion Multiplication

With several ways to represent these quaternions, we also have several ways through which we can represent quaternion multiplication:

1. **Real and 3 Imaginary Parts**:

$$\begin{aligned}(p_0 + p_x i + p_y j + p_z k)((q_0 + q_x i + q_y j + q_z k) =& (p_0 q_0 - p_x q_x - p_y q_y - p_z q_z) + \\ & (p_0 q_x + p_x q_0 + p_y q_z - p_z q_y) i + \\ & (p_0 q_y - p_x q_z + p_y q_z + p_z q_x) j + \\ & (p_0 q_z + p_x q_y - p_y q_x + p_z q_0) k\end{aligned}$$

2. **Scalar and 3-Vector**:

$$(p, \mathbf{p})(q, \mathbf{q}) = (pq - \mathbf{p} \cdot \mathbf{q}, p\mathbf{q} + q\mathbf{p} + \mathbf{p} \times \mathbf{q})$$

   Note: This multiplication operation is not commutative.

3. **4-Vector**:

$$\begin{bmatrix} p_0 & -p_x & -p_y & -p_z \\ p_x & p_0 & -p_z & p_y \\ p_y & p_z & p_0 & -p_x \\ p_z & -p_y & p_x & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{bmatrix}$$

   Note: Here we also have an isomorphism between the quaternion and the 4 x 4 orthogonal matrix (this matrix is orthonormal if we have unit quaternions). Here we can show the isomorphism and relate this back to the cross product we saw before by considering the equivalence of the two following righthand-side expressions:

   (a) $\overset{o}{p}\overset{o}{q} = P\overset{o}{q}$, where $P = \begin{bmatrix} p_0 & -p_x & -p_y & -p_z \\ p_x & p_0 & -p_z & p_y \\ p_y & p_z & p_0 & -p_x \\ p_z & -p_y & p_x & p_0 \end{bmatrix}$

   (b) $\overset{o}{p}\overset{o}{q} = \bar{\mathbf{q}}\overset{o}{p}$, where $\bar{\mathbf{q}} = \begin{bmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & q_z & -q_y \\ q_y & -q_z & q_0 & q_x \\ q_z & q_y & -q_x & q_0 \end{bmatrix}$

   A few notes about these matrices $\mathbf{P}$ and $\bar{\mathbf{q}}$:

   - These matrices are orthonormal if quaternions are unit quaternions
   - $\mathbf{P}$ is normal if $\overset{o}{p}$ is a unit quaternion, and $\bar{\mathbf{q}}$ is normal if $\overset{o}{q}$ is a unit quaternion.
   - $\mathbf{P}$ is skew-symmetric if $\overset{o}{p}$ has zero scalar part, and $\bar{\mathbf{q}}$ is skew-symmetric if $\overset{o}{q}$ has zero scalar part.
   - $\mathbf{P}$ and $\bar{\mathbf{q}}$ have the same signs for the first row and column, and flipped signs for off-diagonal entries in the bottom right 3 x 3 blocks of their respective matrices.

1.6.5 Properties of 4-Vector Quaternions

These properties will be useful for representing vectors and operators such as rotation later:

1. **Not commutative**: $\overset{o}{p}\overset{o}{q} \neq \overset{o}{q}\overset{o}{p}$
2. **Associative**: $(\overset{o}{p}\overset{o}{q})\overset{o}{r} = \overset{o}{p}(\overset{o}{q}\overset{o}{r})$
3. **Conjugate**: $(p, \mathbf{p})^* = (p, -\mathbf{p}) \implies (\overset{o}{p}\overset{o}{q}) = \overset{o}{q}^{*}\overset{o}{p}$
4. **Dot Product**: $(p, \mathbf{p}) \cdot (q, \mathbf{q}) = pq + \mathbf{p} + \mathbf{q}$
5. **Norm**: $||\overset{o}{q}||_2^2 = \overset{o}{q} \cdot \overset{o}{q}$
6. **Conjugate Multiplication**: $\overset{o}{q}\overset{o}{q}^{*}$:

$$\begin{aligned} \overset{o}{q}\overset{o}{q}^{*} &= (q, \mathbf{q})(q, -\mathbf{q}) \\ &= (q^2 + \mathbf{q} \cdot \mathbf{q}, 0) \\ &= (\overset{o}{q} \cdot \overset{o}{q})\overset{o}{e} \end{aligned}$$

   Where $\overset{o}{e} \triangleq (1, 0)$, i.e. it is a quaternion with no vector component. Conversely, then, we have: $\overset{o}{q}^{*}\overset{o}{q} = (\overset{o}{q}\overset{o}{q})\overset{o}{e}$.

7. **Multiplicative Inverse**: $\overset{o}{q}^{-1} = \frac{\overset{o}{q}^{*}}{(\overset{o}{q} \cdot \overset{o}{q})}$ (Except for $\overset{o}{q} = (0, \mathbf{0})$, which is problematic with other representations anyway.)

We can also look at properties with dot products:

1. $(\overset{o}{p}\overset{o}{q}) \cdot (\overset{o}{p}\overset{o}{q}) = (\overset{o}{p} \cdot \overset{o}{p})(\overset{o}{q} \cdot \overset{o}{q})$

2. $(\overset{o}{p}\overset{o}{q}) \cdot (\overset{o}{p}\overset{o}{r}) = (\overset{o}{p} \cdot \overset{o}{p})(\overset{o}{q} \cdot \overset{o}{r})$

3. $(\overset{o}{p}\overset{o}{q}) \cdot \overset{o}{r} = \overset{o}{p} \cdot (\overset{o}{r}\overset{o}{q}^*)$

We can also represent these quaternions as vectors. Note that these quaternions all have zero scalar component.

1. $\overset{o}{r} = (0, \mathbf{r})$
2. $\overset{o}{r}^* = (0, -\mathbf{r})$
3. $\overset{o}{r} \cdot \overset{o}{s} = \mathbf{r} \cdot \mathbf{s}$
4. $\overset{o}{r}\overset{o}{s} = (-\mathbf{r} \cdot \mathbf{s}, \mathbf{r} \times \mathbf{s})$
5. $(\overset{o}{r}\overset{o}{s}) \cdot \overset{o}{t} = \overset{o}{r} \cdot (\overset{o}{s}\overset{o}{t}) = [\mathbf{r}\ \mathbf{s}\ \mathbf{t}]$ (Triple Products)
6. $\overset{o}{r}\overset{o}{r} = -(\mathbf{r} \cdot \mathbf{r})\overset{o}{e}$

Another **note**: For representing rotations, we will use unit quaternions. We can represent scalars and vectors with:

- **Representing scalars**: $(s, \mathbf{0})$
- **Representing vectors**: $(0, \mathbf{v})$

1.7 Quaternion Rotation Operator

To represent a rotation operator using quaternions, we need a quaternion operation that maps from vectors to vectors. More specifically, we need an operation that maps from 4D, the operation in which quaternions reside, to 3D in order to ensure that we are in the correct for rotation in 3-space. Therefore, our rotation operator is given:

$$\begin{aligned}\overset{o}{r}' = R(\overset{o}{r}) &= \overset{o}{q}\overset{o}{r}\overset{o}{q}^* \\ &= (Q\overset{o}{r})\overset{o}{q}^* \\ &= (\bar{Q}^T Q)\overset{o}{r}\end{aligned}$$

Where the matrix $\bar{Q}^T Q$ is given by:

$$\bar{Q}^T Q = \begin{bmatrix} \overset{o}{q} \cdot \overset{o}{q} & 0 & 0 & 0 \\ 0 & q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_0 q_z) & 2(q_x q_z + q_0 q_y) \\ 0 & 2(q_y q_x + q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_0 q_x) \\ 0 & 2(q_z q_x - q_0 q_y) & 2(q_z q_y + q_0 q_z) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix}$$

A few notes about this matrix:

- Since the scalar component of $\overset{o}{q}$ is zero, the first row and matrix of this column are sparse, as we can see above.
- If $\overset{o}{q}$ is a unit quaternion, the lower right 3×3 matrix of $\bar{Q}^T Q$ will be orthonormal (it is an orthonormal rotation matrix).

Let us look at more properties of this mapping $\overset{o}{r}' = \overset{o}{q}\overset{o}{r}\overset{o}{q}^*$:

1. **Scalar Component**: $r' = r(\overset{o}{q} \cdot \overset{o}{q})$
2. **Vector Component**: $\mathbf{r}' = (q^2 - \mathbf{q} \cdot \mathbf{q})\mathbf{r} + 2(\mathbf{q} \cdot \mathbf{r})\mathbf{q} + 2q(\mathbf{q} \times \mathbf{r})$
3. **Operator Preserves Dot Products**: $\overset{o}{r}' \cdot \overset{o}{s}' = \overset{o}{r} \cdot \overset{o}{s} \implies \mathbf{r}' \cdot \mathbf{s}' = \mathbf{r} \cdot \mathbf{s}$
4. **Operator Preserves Triple Products**: $(\overset{o}{r}' \cdot \overset{o}{s}') \cdot \overset{o}{t}' = (\overset{o}{r} \cdot \overset{o}{s}) \cdot \overset{o}{t} \implies (\mathbf{r}' \cdot \mathbf{s}')\mathbf{t}' = (\mathbf{r} \cdot \mathbf{s}) \cdot \mathbf{t} \implies [\mathbf{r}'\ \mathbf{s}'\ \mathbf{t}'] = [\mathbf{r}\ \mathbf{s}\ \mathbf{t}]$
5. **Composition (of rotations!)**: Recall before that we could not easily compose rotations with our other rotation representations. Because of associativity, however, we can compose rotations simply through quaternion multiplication:

   $$\overset{o}{p}(\overset{o}{q}\overset{o}{r}\overset{o}{q}^*)\overset{o}{p}^* = (\overset{o}{p}\overset{o}{q})\overset{o}{r}(\overset{o}{q}^*\overset{o}{p}^*) = (\overset{o}{p}\overset{o}{q})\overset{o}{r}(\overset{o}{p}\overset{o}{q})^*$$

   I.e. if we denote the product of quaternions $\overset{o}{z} \triangleq \overset{o}{p}\overset{o}{q}$, then we can write this rotation operator as a single rotation:

   $$\overset{o}{p}(\overset{o}{q}\overset{o}{r}\overset{o}{q}^*)\overset{o}{p}^* = (\overset{o}{p}\overset{o}{q})\overset{o}{r}(\overset{o}{q}^*\overset{o}{p}^*) = (\overset{o}{p}\overset{o}{q})\overset{o}{r}(\overset{o}{p}\overset{o}{q})^* = \overset{o}{z}\overset{o}{r}\overset{o}{z}^*$$

   This ability to compose rotations is quite advantageous relative to many of the other representations of rotations we have seen before (orthonormal rotation matrices can achieve this as well).

1.7.1 Relation of Quaternion Rotation Operation to Rodrigues Formula

Let us see how this operator relates to the Rodrigues formula and the axis-angle representation of rotations:

$$\begin{aligned}\mathbf{r}' &= (q^2 - \mathbf{q}\cdot\mathbf{q})\mathbf{r} + 2(\mathbf{q}\cdot\mathbf{r})\mathbf{q} + 2q(\mathbf{q}\times\mathbf{r}) \\ &= (\cos\theta)\mathbf{r} + (1-\cos\theta)(\hat{\boldsymbol{\omega}}\cdot\mathbf{r})\hat{\boldsymbol{\omega}} + (\sin\theta)(\hat{\boldsymbol{\omega}}\times\mathbf{r})\end{aligned}$$

We have that q is parallel to $\hat{\boldsymbol{\omega}}$. Note the following equalities:

- $(q^2 - \mathbf{q}\cdot\mathbf{q}) = \cos\theta$
- $2||\mathbf{q}||_2^2 = (1-\cos\theta)$
- $2q||\mathbf{q}||_2^2 = \sin\left(\frac{\theta}{2}\right)$
- $q = \cos\left(\frac{\theta}{2}\right)$
- $||\mathbf{q}||_2^2 = \sin\left(\frac{\theta}{2}\right)$

From these statements of equality, we can conclude:

$$\overset{o}{q} = (\cos\big(\frac{\theta}{2}\big), \hat{\boldsymbol{\omega}}\sin\big(\frac{\theta}{2}\big))$$

A few notes on this:

- We see that both the scalar and vector components of the quaternion $\overset{o}{q}$ depend on the axis of rotation $\hat{\boldsymbol{\omega}}$ and the angle of rotation θ.
- The vector component of this quaternion is parallel to $\hat{\boldsymbol{\omega}}$.
- This representation is one way to represent a unit quaternion.
- Knowing the axis and angle of a rotation allows us to compute the quaternion.
- Note that $-\overset{o}{q}$ represents the same mapping as $\overset{o}{q}$ since:

$$(-\overset{o}{q})\overset{o}{r}(-\overset{o}{q}^*) = \overset{o}{q}\overset{o}{r}\overset{o}{q}^*$$

To build intuition with this quaternion rotation operator, one way we can conceptualize this is by considering that our space of rotations is a 3D sphere in 4D, and opposite points on this sphere represent the same rotation.

1.8 Applying Quaternion Rotation Operator to Photogrammetry

Now that we have specified this operator and its properties, we are ready to apply this to photogrammetry, specifically for **absolute orientation**. Let us briefly review our four main problems of photogrammetry:

1. **Absolute Orientation** (3D to 3D): Range Data
2. **Relative Orientation** (2D to 2D): Binocular Stereo
3. **Exterior Orientation** (3D to 2D): Passive Navigation
4. **Interior Orientation** (2D to 3D): Camera Calibration

We will start by applying this to our first problem, **absolute orientation**. Recall our goal with this photogrammetry problem is to find the 3D transformation between our coordinate systems.

1.8.1 Least Squares Approach to Find R

Recall our first attempt to solve for this transformation was done through least squares to solve for an optimal rotation matrix and a translation vector. While we showed in lecture 17 that we can solve for an optimal translation vector that depends on the rotation matrix R, we could not find a closed-form solution for the rotation matrix R. We review why:

$$\begin{aligned}\min +R\sum_{i=1}^{n}||\mathbf{e}_i||_2^2 &= \sum_{i=1}^{n}(\mathbf{r}'_{r,i} - R(\mathbf{r}'_{l,i})(\mathbf{r}'_{r,i} - R(\mathbf{r}'_{l,i})) \\ &= \sum_{i=1}^{n}||\mathbf{r}'_{r,i}||_2^2 - \sum_{i=1}^{n}(\mathbf{r}'_{r,i} - R(\mathbf{r}'_{l,i})) - \sum_{i=1}^{n}||\mathbf{r}'_{l,i}||_2^2\end{aligned}$$

Where the first of these terms is fixed, the second of these terms is to be maximized (since there is a negative sign in front of this term, this thereby minimizes the objective), and the third of these terms is fixed. Therefore, our rotation problem can be simplified to:

$$R^* = \min_R \sum_{i=1}^{n}||\mathbf{e}_i||_2^2 = -2\sum_{i=1}^{n}(\mathbf{r}'_{r,i} - R(\mathbf{r}'_{l,i})) \tag{1}$$

$$= \min_R(-2\sum_{i=1}^{n}\mathbf{r}'_{r,i} \cdot R(\mathbf{r}'_{l,i})) \tag{2}$$

$$= \max_R(\sum_{i=1}^{n} rb'_{r,i} \cdot R(\mathbf{r}'_{l,i})) \tag{3}$$

But since we are optimizing over an orthonormal rotation matrix R, we cannot simply take the derivative and set it equal to zero as we usually do for these least squares optimization problems. Though we can solve this as a Lagrangian optimization problem, specifying these constraints is difficult and makes for a much more difficult optimization problem. It turns out this a common problem in spacecraft attitude control. Let us see how we can use quaternions here!

1.8.2 Quaternion-based Optimization

Solving this with our quaternion operator above, and noting the following definitions:

- $\overset{o}{r}{}'_{l,i} = (0, \mathbf{r}'_{l,i})$
- $\overset{o}{r}{}'_{r,i} = (0, \mathbf{r}'_{l,i})$

Then we can solve for our optimal rotation by solving for quaternions instead:

$$\begin{aligned}R^* &= \max_R \sum_{i=1}^{n} rb'_{r,i} \cdot R(\mathbf{r}'_{l,i}) \\ &= \max_{\overset{o}{q}, ||\overset{o}{q}||_2=1} \sum_{i=1}^{n}(\overset{o}{q}\overset{o}{r}{}'_{l,i}\overset{o}{q}{}^*) \cdot \overset{o}{r}{}'_{r,i} \\ &= \max_{\overset{o}{q}, ||\overset{o}{q}||_2=1} \sum_{i=1}^{n}(\overset{o}{q}\overset{o}{r}{}'_{l,i}) \cdot (\overset{o}{r}{}'_{r,i}\overset{o}{q}) \\ &= \max_{\overset{o}{q}, ||\overset{o}{q}||_2=1} \sum_{i=1}^{n}(\bar{R}_{l,i}\overset{o}{q}) \cdot (R_{r,i}\overset{o}{q}) \\ &= \max_{\overset{o}{q}, ||\overset{o}{q}||_2=1} \overset{o}{q}{}^T\Big(\sum_{i=1}^{n}\bar{R}_{l,i}^T R_{r,i}\Big)\overset{o}{q} \quad \textbf{(Since } \overset{o}{q} \textbf{ does not depend on } i\textbf{)}\end{aligned}$$

Where the term in the sum is a 4×4 matrix derived from point cloud measurements.

From here, we can solve for an optimal rotation quaternion through Lagrangian Optimization, with our objective given by:

$$\max_{\overset{o}{q}} \overset{o}{q}{}^T N\overset{o}{q}, \text{ subject to: } \overset{o}{q} \cdot \overset{o}{q} = 1,\ N \triangleq \sum_{i=1}^{n}\bar{R}_{l,i}^T R_{r,i}$$

Then written with the Lagrangian constraints this optimization problem becomes:

$$\max_{\overset{o}{q}} \overset{o}{q}^T N \overset{o}{q} + \lambda(1 - \overset{o}{q} \cdot \overset{o}{q})$$

Differentiating this expression w.r.t. $\overset{o}{q}$ and setting the result equal to zero yields the following first-order condition:

$$2N\overset{o}{q} - 2\lambda\overset{o}{q} = 0$$

It turns out that similarly to our inertia problem from the last lecture, the quaternion solution to this problem is the solution to an eigenvalue/eigenvector problem. Specifically, our solution is the eigenvector corresponding to the largest eigenvalue of a 4×4 real symmetric matrix N constructed from elements of the matrix given by a dyadic product of point cloud measurements from the left and righthand systems of coordinates:

$$M = \sum_{i=1}^{n} \mathbf{r}'_{l,i} \mathbf{r}'_{r,i}{}^T$$

This matrix M is an asymmetric 3×3 real matrix. A few other notes about this:

- The **eigenvalues** of this problem are the **Lagrange Multipliers** of this objective, and the eigenvectors are derived from the eiegenvectors of N, our matrix of observations.
- This analytic solution leads to/requires solving a quartic polynomial - fortunately, we have closed-form solutions of polynomials up to a quartic degree! Therefore, a closed-form solution exists. Specifically, the characteristic equation in this case takes the form:

  $$\lambda^4 + c_3\lambda^3 + c_2\lambda^2 + c_1\lambda + c_0 = 0$$

  Because the matrix of point cloud measurements N is symmetric, this characteristic equation simplifies and we get the following coefficients:

  1. $c_3 = \text{tr}(N) = 0$
  2. $c_2 = -2\text{tr}(M^T M)$
  3. $c_1 = -8 \det|M|$
  4. $c_0 = \det|N|$

In addition to solving **absolute orientation** problems with quaternions, this approach has applications to other problems as well, such as:

- Relative Orientation (Binocular Stereo)
- Camera Calibration
- Manipulator Kinematics
- Manipulator Fingerprinting
- Spacecraft Dynamics

1.9 Desirable Properties of Quaternions

Next, let us look at what propreties we would like to obtain from using quaternions, and let us see how well these properties are satisfied:

- **The ability to rotate a vector or coordinate system**
  Yes!
- **Ability to compose rotations**
  Yes!
- **Intuitive, non-redundant representations**
  There is some redundancy (but note that this is solved by using unit quaternions!), and this representation is not the most intuitive.

- **Computational efficiency**
  We will discuss this in more detail further below.
- **Ability to Interpolate Orientations**
  Yes!
- **Ability to average over rotations**
  Yes!
- **Ability to take derivative w.r.t. rotation - e.g. for optimization and least squares**
  Yes!
- **Ability to sample rotations**
  Yes!
- **Notion of a space of rotations**
  Yes!

1.9.1 Computational Issues for Quaternions

One helpful metric we can compare different representations for rotation on is their computational efficiency. Below, we compare quaternions for rotation with orthonormal matrices for rotation in several important operations:

- **Operation: Composition of Rotations**: $\overset{o}{p}\overset{o}{q}$
  This is given by:

  $$\overset{o}{p}\overset{o}{q} = (p, \mathbf{p})(q, \mathbf{q}) = (pp - \mathbf{p}\mathbf{q}, p\mathbf{q} + q\mathbf{p} + \mathbf{p} \times \mathbf{q})$$

  Carrying this out naively requires **16 multiplications and 12 additions**.

  Compared with orthonormal matrices, composing quaternions is **faster** for this operation (orthonormal matrices require **27 multiplications and 18 additions**.

- **Operation: Rotating Vectors**: $\overset{o}{q}\overset{o}{r}\overset{o}{q}^*$
  This is given by:

  $$\begin{aligned}\overset{o}{q}\overset{o}{r}\overset{o}{q}^* \rightarrow & \mathbf{r}' = (q^2 - \mathbf{r}\cdot\mathbf{q})\mathbf{r} + 2(\mathbf{q}\cdot\mathbf{r})\mathbf{q} + 2q(\mathbf{q}\times\mathbf{r}) \\ & \mathbf{r}' = \mathbf{r} + 2q(\mathbf{q}\times\mathbf{r}) + 2\mathbf{q}\times(\mathbf{q}\times\mathbf{r}) \ \ \textbf{(More efficient implementation)}\end{aligned}$$

  Carrying this out naively requires **15 multiplications and 12 additions**.

  Compared with orthonormal matrices, composing quaternions is **slower** for this operation (orthonormal matrices require **9 multiplications and 6 additions**.

- **Operation: Renormalization** This operation is used when we compose many rotations, and the quaternion (if we are using a quaternion) or the orthonormal matrix is not quite an orthonormal matrix due to floating-point arithmetic. Since this operation requires matrix inversion (see below) for orthonormal matrices, it is much faster to carry out this operation with quaternions.

  Nearest Unit Quaternion: $\frac{\overset{o}{q}}{\sqrt{\overset{o}{q}\cdot\overset{o}{q}}}$

  Nearest Orthonormal Matrix: $M(M^T M)^{-\frac{1}{2}}$

1.9.2 Space of Rotations

We will conclude today's lecture by discussing the space of rotations, now that we have a representation for it:

- S^3 (the unit sphere $\in \mathbb{R}^3$) with antipodal points identified
- Projective space P^3
- **Sampling**: Sampling in this space can be done in regular and random intervals

- **Finite rotation groups**: These include the platonic solids with 12, 24, 60 elements - we can have rotation groups for (i) tetrahedron, (ii) hexahedron/octahedron, and (iii) dodecahedron/icosahedron
- **Finer-grade Sampling** can be achieved by sub-dividing the simplex in the rotation space
- If $\{\overset{o}{q}_i\}_{i=1}^N$ is a group, then so is $\{\overset{o}{q}'_i\}_{i=1}^N$, where $\overset{o}{q}'_i = \overset{o}{q}_0 \overset{o}{q}_i$.

1.10 References

1. Group, https://en.wikipedia.org/wiki/Group_(mathematics)
2. Manifold, https://en.wikipedia.org/wiki/Manifold