CS612 - Algorithms in Bioinformatics

Structural Manipulation

April 3, 2023

Rapid Structural Analysis Methods

- Emergence of large structural databases which do not allow manual (visual) analysis and require efficient 3-D search and classification methods.
- Structure is much better preserved than sequence – proteins may have similar structures but dissimilar sequences.
- Structural motifs may predict similar biological function
- Getting insight into protein folding. Recovering the limited (?) number of protein folds.
- Comparing proteins of not necessarily the same family.

Rapid Structural Analysis Methods

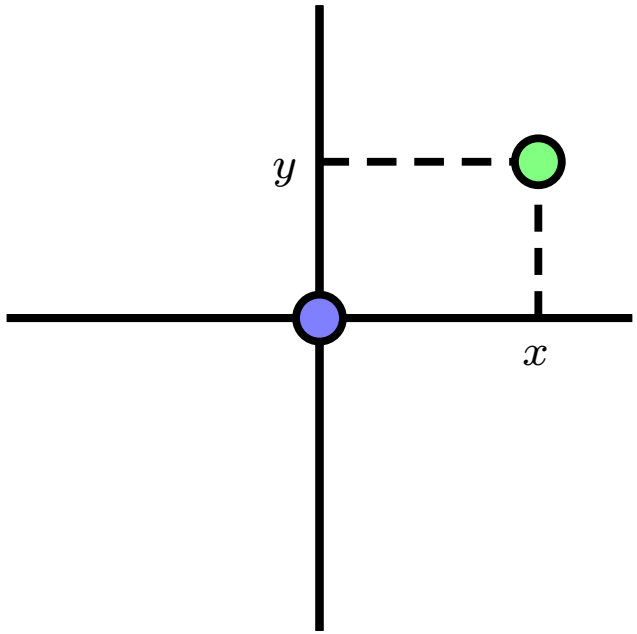
- Implementing structural algorithms (folding, docking, alignment) requires geometric manipulation of protein structures.
- A 3-D protein structure is represented as a set of x, y, z coordinates (vectors).
- Structural manipulation is done via *geometric transformations* (translation, rotation) of some or all the coordinates.
- Transformations can be represented using matrices applied on the coordinate vectors.

Degrees of Freedom (DOFs)

Definition (Degrees of Freedom)

The degree of freedom (DOF) is the set of independent parameters that can be varied to define the state of the system

Examples:
The location of a point in a 2-D cartesian system has two independent parameters – its (x, y) coordinates.



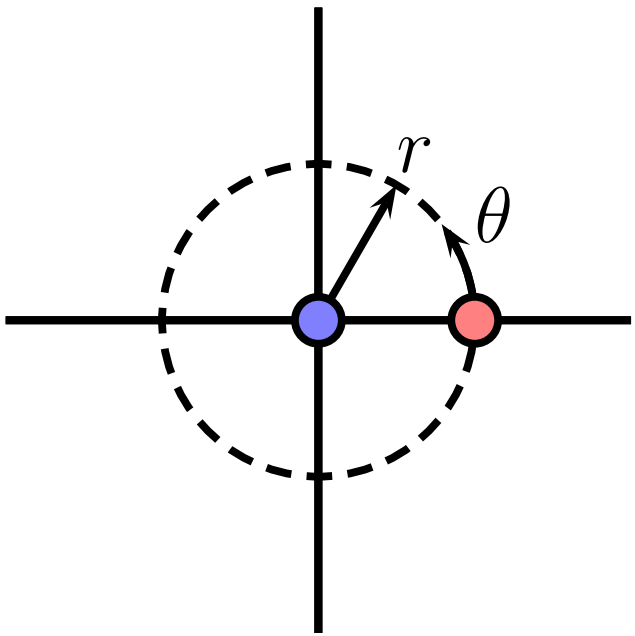
Degrees of Freedom (DOFs)

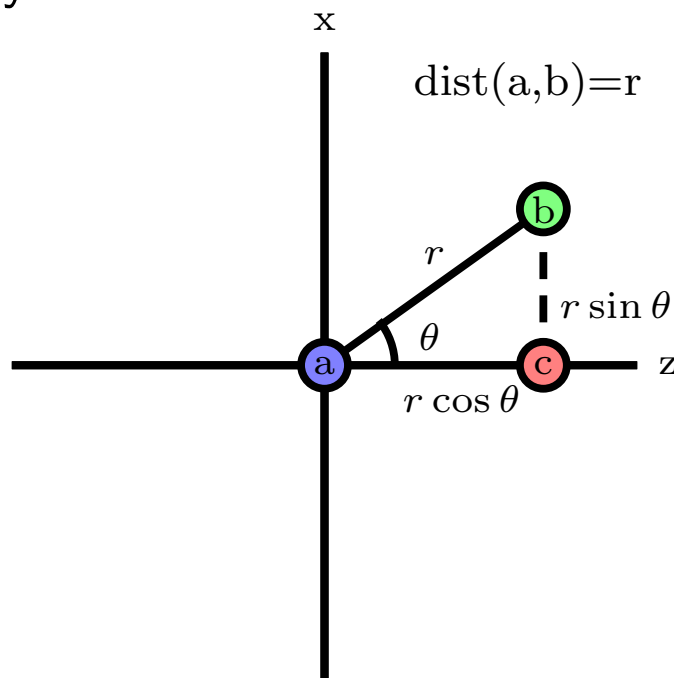
Definition (Degrees of Freedom)

The degree of freedom (DOF) is the set of independent parameters that can be varied to define the state of the system

Examples:

An alternative representation – (r, θ), distance from the origin and rotation about the origin, respectively.





Degrees of Freedom (DOFs)

Definition (Degrees of Freedom)

The degree of freedom (DOF) is the set of independent parameters that can be varied to define the state of the system

Examples:

- The location of a point in a 2-D cartesian system has three independent parameters – its (x, y) coordinates.
- An alternative representation – (r, θ), distance from the origin and rotation about the origin, respectively.
- A molecule with n atoms can be represented by a set of $3 \times N$ cartesian coordinates, so it has $3 \times N$ DOFs...
- Or does it?
- The actual number of DOFs is smaller, since distance and angle constraints restrict the atomic movement.

Representation by Internal Coordinates

- When trying to manipulate the structure internal coordinates may be easier to work with.
- The internal coordinates represent bond length, angles and dihedrals.
- Remember that we treat bond lengths and planar angles as fixed, but we still need them.
- They help us infer the connectivity of the structure and switch between representations.

Representation by Internal Coordinates

- Representing protein conformations with the dihedral angles as the only underlying degrees of freedom is known as the **idealized** or rigid geometry model.
- Ignoring bond lengths and bond angles greatly reduces the number of degrees of freedom and therefore the computational complexity of representing and manipulating protein structures.

Representation by Internal Coordinates

- As a reminder – there are two freely rotatable backbone dihedral angles per amino acid residue in the protein chain: ϕ is a consequence of the rotation about the bond between N and Cα, and ψ , which is a consequence of the rotation about the bond between Cα and C.
- The peptide bond between C of one residue and N of the adjacent residue is not rotatable.
- The number of backbone dihedrals per amino acid is 2 (except the first and last), a total of 2N-2.
- but the number of side chain dihedrals varies with the length of the side chain. Its value ranges from 0, in the case of glycine, which has no side chain, to 5 in the case of arginine.

Representation by Internal Coordinates

- One can generate different three dimensional structures of the same protein by varying the dihedral angles.
- There are 2N-2 backbone dihedral DOFs for a protein with N amino acids, and up to 4N side chain dihedrals that one can vary to generate new protein conformations.
- Changes in backbone dihedral angles generally have a greater effect on the overall shape of the protein than changes in side chain dihedral angles (why?)

Manipulation of Molecular Structures

- When generating new conformations by manipulating the dihedral angles, we will normally require a way to modify the Cartesian coordinates when dihedral rotations are performed, to reflect the new atomic positions.
- This can be easily done with **rotation matrices**.
- Therefore we will probably need to keep the two representations simultaneously.

Rotation Matrices and Translation Vectors

- An $N \times N$ matrix R is a rotation matrix in dimension N if it is orthonormal (its columns are pairwise orthogonal and normalized) and $det(R) = 1$.
- Such matrix has the property $R^T = R^{-1}$.
- A vector $t = \{t_1, t_2, ..., t_N\}$ is a translation vector in dimension N.
- A rigid transformation on a vector v (rotation + translation) has the form $Rv + t$
- Notice that a rotation is followed by a translation and not the other way around.
- Rotation matrices are a *group* under matrix multiplication.

Homogeneous Coordinates

- A translation is not a *linear transformation*, since it does not keep the origin fixed.
- However, it is an *affine transformation* that does not bend or twist its input: lines have to stay linear, parallel lines have to stay parallel and planes have to stay planar.
- Therefore, a rotation matrix and a translation vector can be combined into rigid affine transformation using *homogeneous coordinates*.
- This is done by adding a "dummy" zero vector to the rotation component and 1 to the translation component.
- In other words, we add a dimension to the matrix, so now it is of the form:

$$T = \left[\begin{array}{c|c} R & t \\ \hline 0 & 1 \end{array}\right]$$

Homogeneous Coordinates

- By doing this we transform the system from the Euclidean space to the *Projective space*.
- The new translation vector, $\{t_1, t_2, \ldots, t_N, 1\}$ is the representation of t in the projective plane rather than the Euclidean plane.
- The added component is a scaling factor which in principle can be any non-zero number, so for convenience we will use 1.
- A 0 would indicate a "point at infinitiy".
- To transform a vector $v = \{v_1, v_2, \ldots, v_n\}$ using homogenous coordinates we simply use $v = \{v_1, v_2, \ldots, v_n, 1\}$ so that we can apply the transformation in $N + 1$ dimensions.
- To transform a point back from the projective plane into Euclidean coordinates we simply ignore the 1.

Rotation Matrices and Groups

- Group – A set G with an operation defined on it.
- 4 defining axioms:
  1. Closure: $\forall a, b \in G, a \circ b \in G$
  2. Associativity: $\forall a, b, c \in G, (a \circ b) \circ c = a \circ (b \circ c)$
  3. Identity: $\exists e \in G \text{s.t} \forall a \in G, a \circ e = e \circ a = a$
  4. Inverse: $\forall a \in G \exists a^{-1} \in G$ s.t $\forall a \in G, a \circ a^{-1} = e$
- Rotation matrices are groups under matrix multiplication.
  1. Closure: The multiplication of every two rotation matrices is a rotation matrix.
  2. Associativity: True for every matrix.
  3. Identity: The identity matrix, which is a rotation by 0 degrees.
  4. Inverse: Rotation in the inverse direction. Multiplying a rotation by its inverse gives the identity.

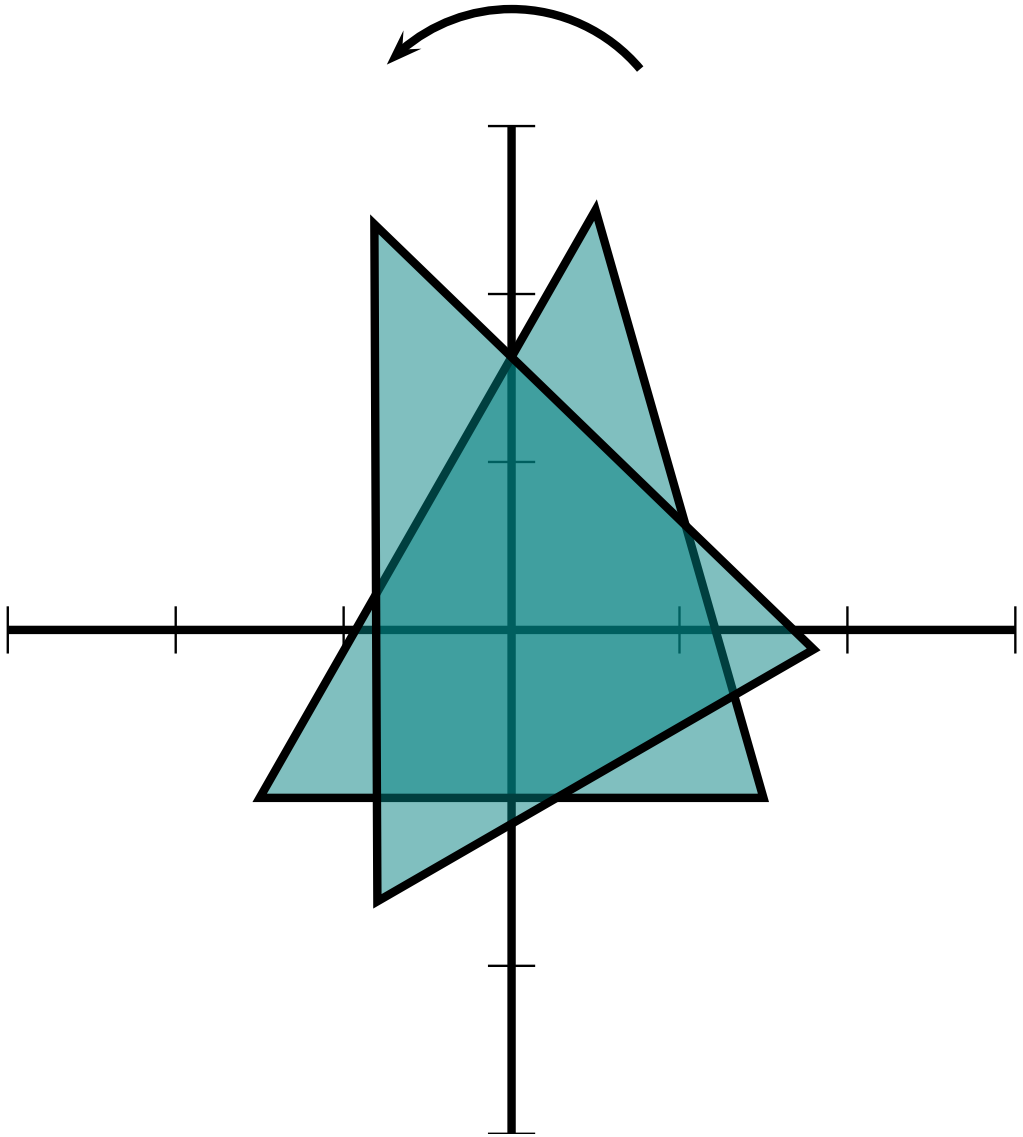
Transformation Matrices

- Objects undergo transformations in space – translation, rotation in 2D or 3D.
- Matrices can encode transformations
- Translation vectors, rotation matrices.
- Example – Rotation in 2D:

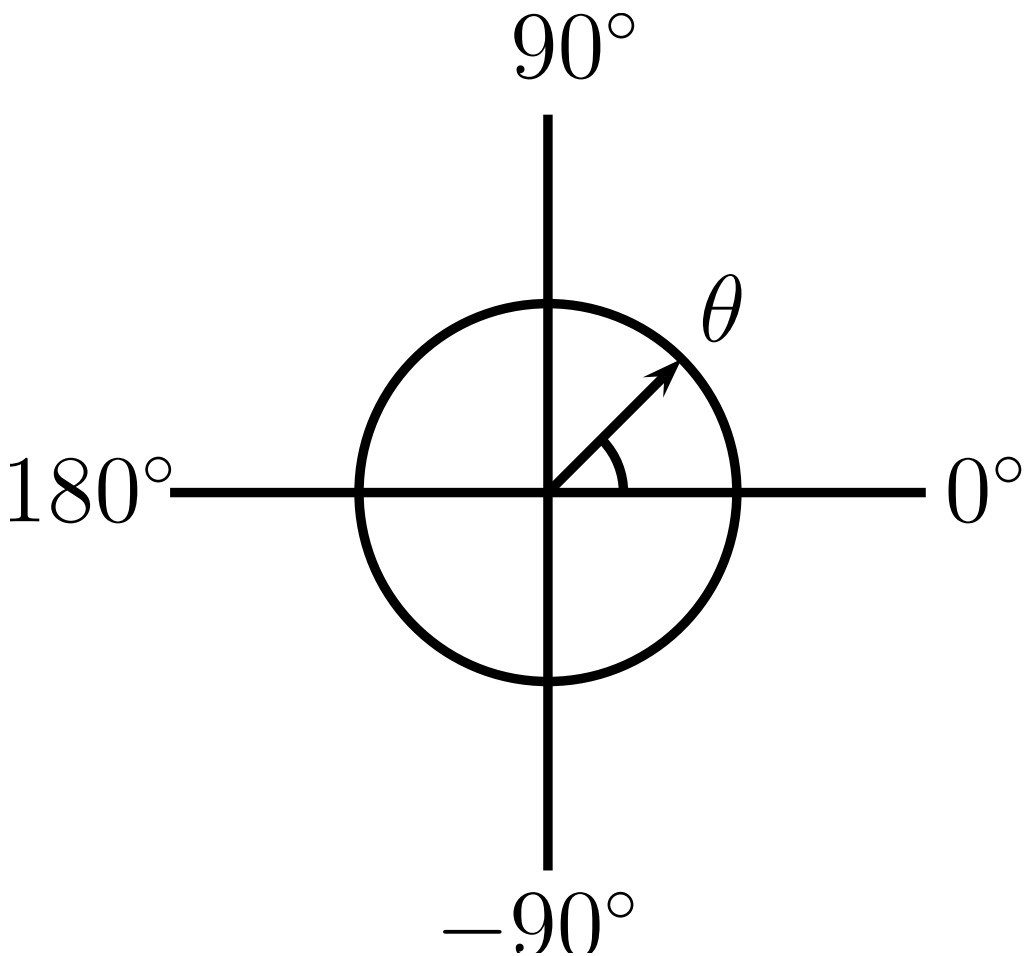
$$A = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

- θ is the rotation angle in the 2D plane.

Example – Rotation in 2D



Representing Rotations in 2D – SO(2)



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

Combining Rotations using Euler's Representation

Matrix multiplication $\leftrightarrow$ unit complex multiplication.

$$\begin{array}{ccc} R(\Theta 1) * R(\Theta 2) & & R(\Theta 1 + \Theta 2) \\ \downarrow & & \uparrow \\ e^{i\Theta 1} * e^{i\Theta 2} & \rightarrow & e^{i(\Theta 1 + \Theta 2)} \end{array}$$

$$a + bi \leftrightarrow \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \simeq S^1$$

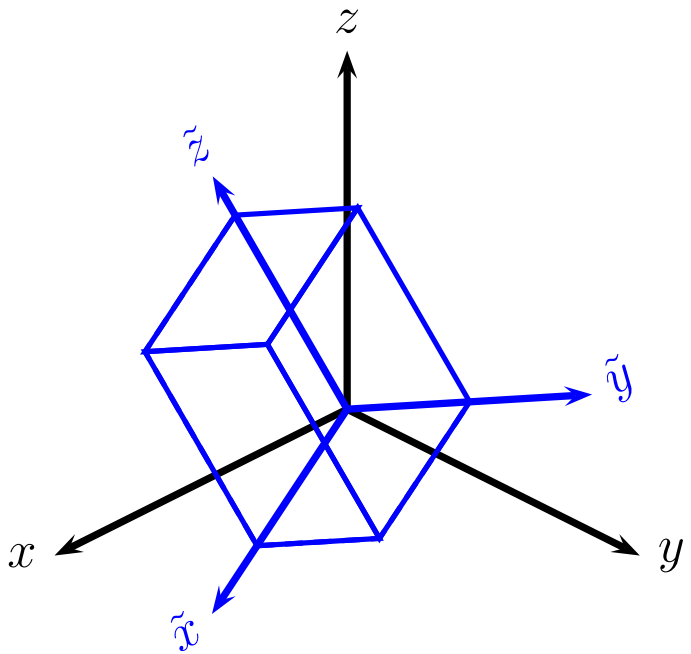
Representing Rotations in 3D

- $SO(3)$ – special orthogonal group in 3D, rigid body rotation in 3D.
- Not a simple extension of 2D rotation.
- How many degrees of freedom are there in $SO3$?
- $O(3) \rightarrow AA^T = 1$.
- 3 constraints on unit row vectors, 3 constraints on orthogonality.
- $SO(3) \rightarrow det(A) = 1$.
- When $det(A) = -1$ it is a reflection.

Representing Rotations in 3D

- 3x3 matrix
- Euler angles (phi,theta,psi)
- Yaw, pitch, roll angles
- Axis-angle representation
- Quaternions

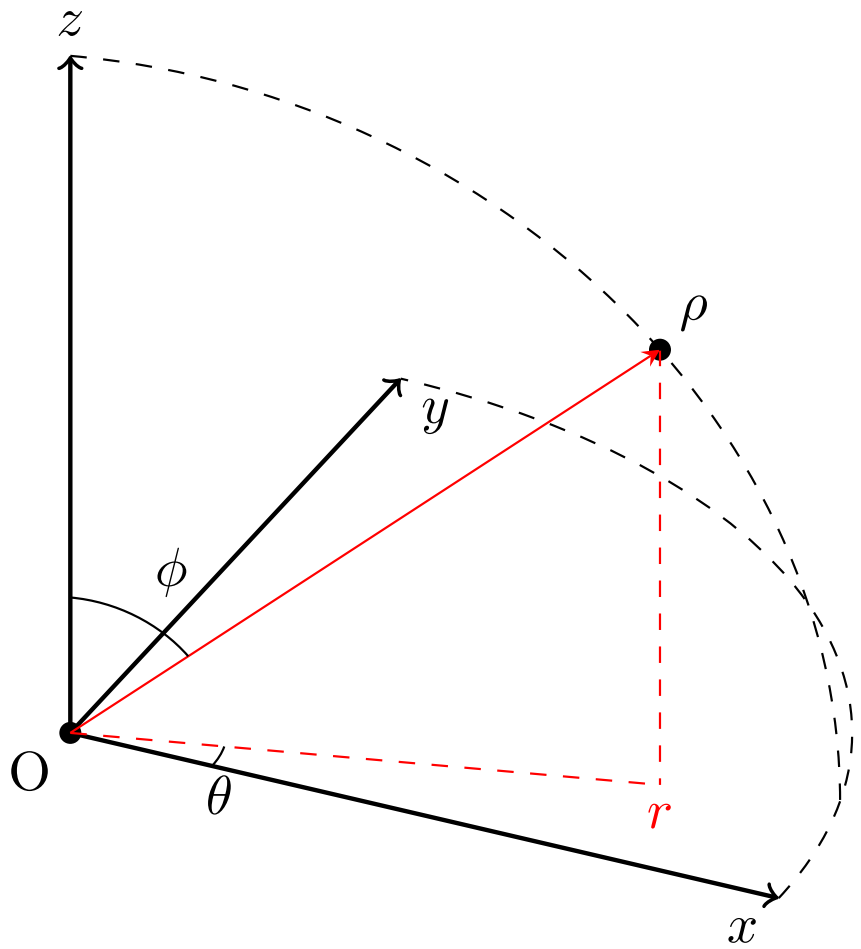
3X3 Matrix



$$R = \begin{bmatrix} \tilde{x}_1 & \tilde{y}_1 & \tilde{z}_1 \\ \tilde{x}_2 & \tilde{y}_2 & \tilde{z}_2 \\ \tilde{x}_3 & \tilde{y}_3 & \tilde{z}_3 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \in SO(3)$$

Spherical Coordinates in 3D

- An extension of polar coordinates.
- ρ – the magnitude of the vector.
- r – the projection of the vector on the xy plane
- θ – same as in polar coordinates w.r.t r on the xy plane.
- ϕ – angle around the z axis.



Performing a Rotation on a Point

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} R_{11}p_x + R_{12}p_y + R_{13}p_z \\ R_{21}p_x + R_{22}p_y + R_{23}p_z \\ R_{31}p_x + R_{32}p_y + R_{33}p_z \end{bmatrix}$$

CCW Rotations Around Axes

- To combine rotations around axes multiply matrices
- Notice that matrix multiplication is not commutative (order matters!)
- Look at order from right to left. For example – $R_x(\gamma)R_y(\beta)$ rotates by *beta* around y and then rotates the result around x by γ

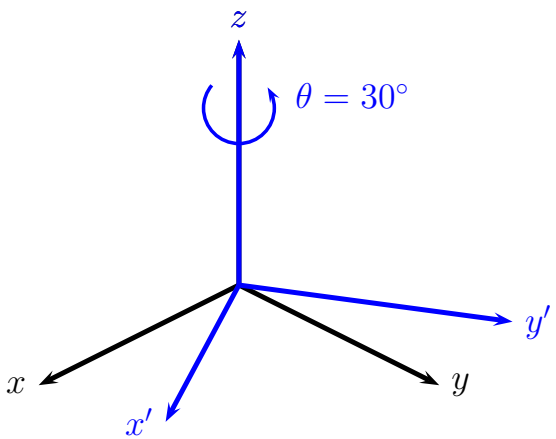
$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$$

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

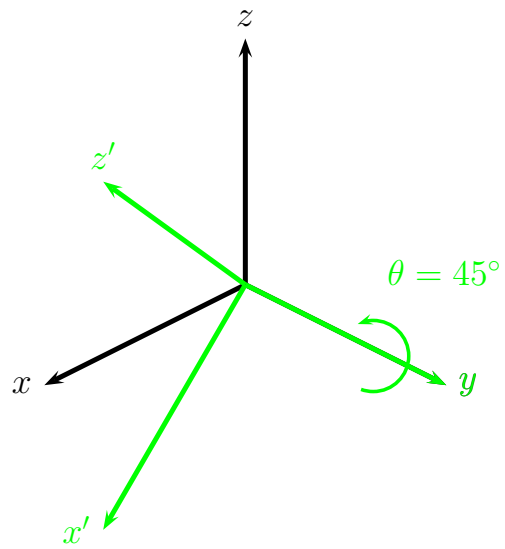
$$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

CCW Rotations Around Axes – Example

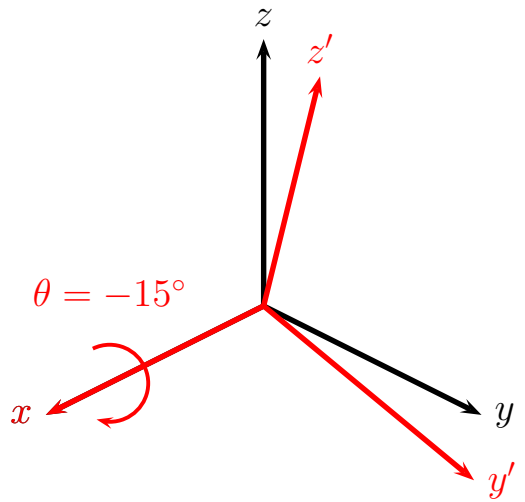
Rotation by 30 degrees around z



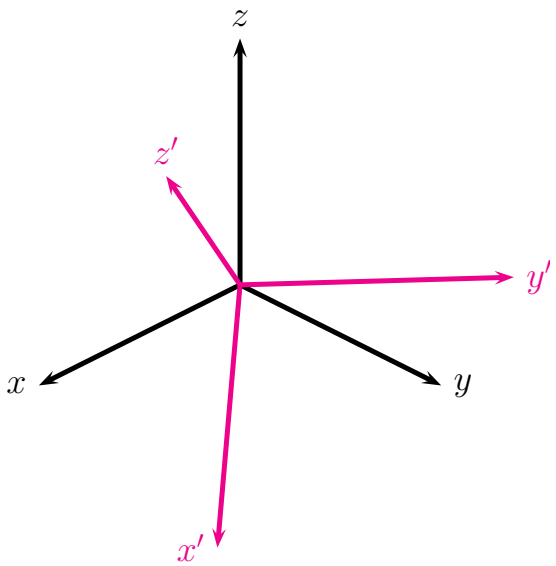
Rotation by 45 degrees around y



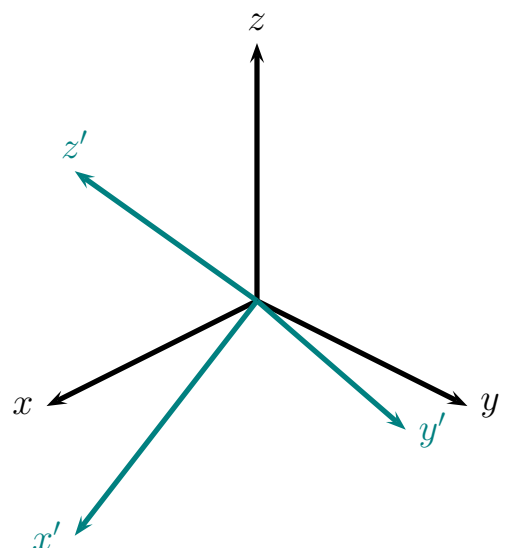
Rotation by -15 degrees around x



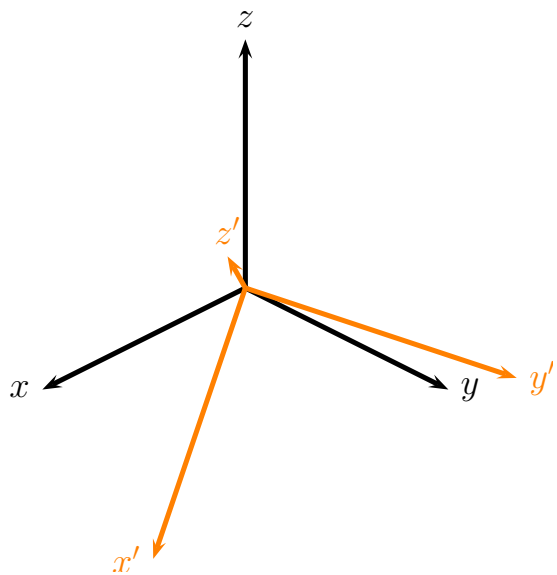
Rotation Sequence – xzy



Rotation sequence – xyz



Rotation Sequence – yzx



CCW Rotations Around Axes – Numerical Example

Let us rotate the vector $v = \{1, 2, 3\}$ around the z axis by 60° and then around the y axis by -60°:

$$\begin{bmatrix} \cos 60 & -\sin 60 & 0 \\ \sin 60 & \cos 60 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} =$$

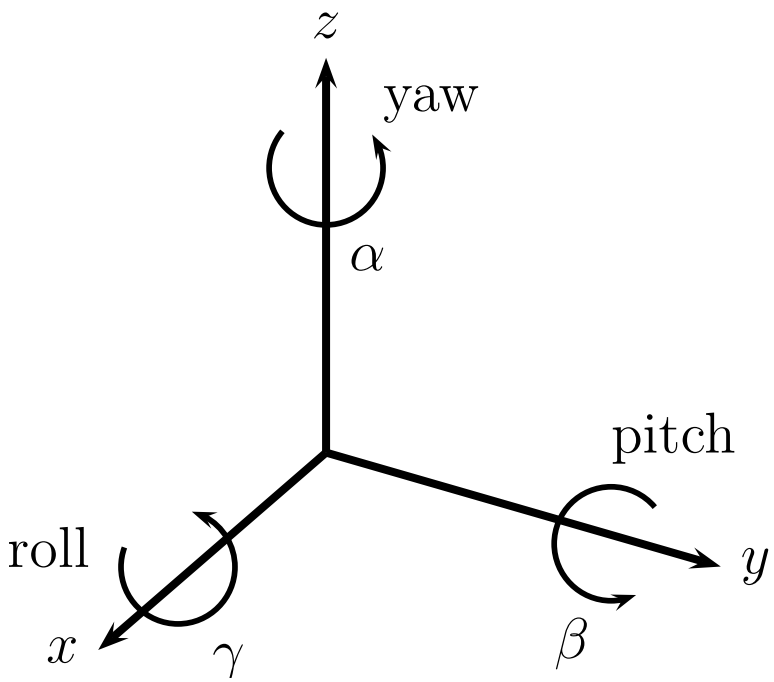
$$\begin{bmatrix} 1 * 0.5 - 2 * 0.866 + 3 * 0 \\ 1 * 0.866 + 2 * 0.5 + 3 * 0 \\ 0 + 0 + 3 * 1 \end{bmatrix} = \begin{bmatrix} -1.232 \\ 1.866 \\ 3 \end{bmatrix}$$

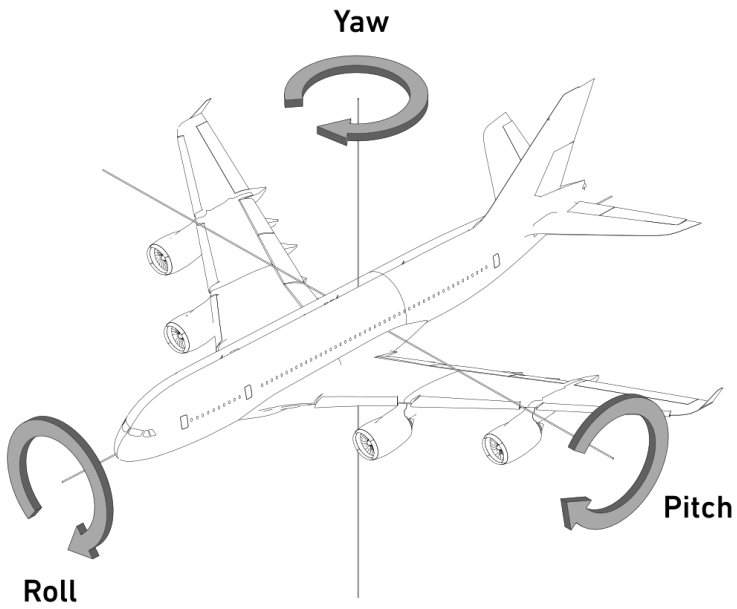
Then:

$$\begin{bmatrix} \cos 60 & 0 & -\sin 60 \\ 0 & 1 & 0 \\ \sin 60 & 0 & \cos 60 \end{bmatrix} \begin{bmatrix} -1.232 \\ 1.866 \\ 3 \end{bmatrix} =$$

$$\begin{bmatrix} -1.232 * 0.5 + 1.866 * 0 - 3 * 0.866 \\ -1.232 * 0 + 1.866 * 1 + 3 * 0 \\ -1.232 * 0.866 + 1.866 * 0 + 3 * 0.5 \end{bmatrix} = \begin{bmatrix} -3.214 \\ 1.866 \\ 0.433 \end{bmatrix}$$

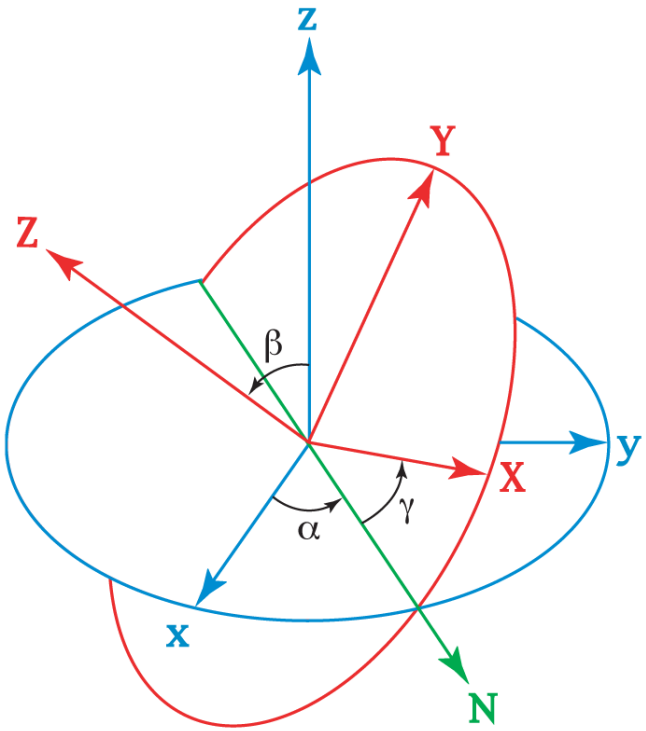
Roll, Pitch, Yaw





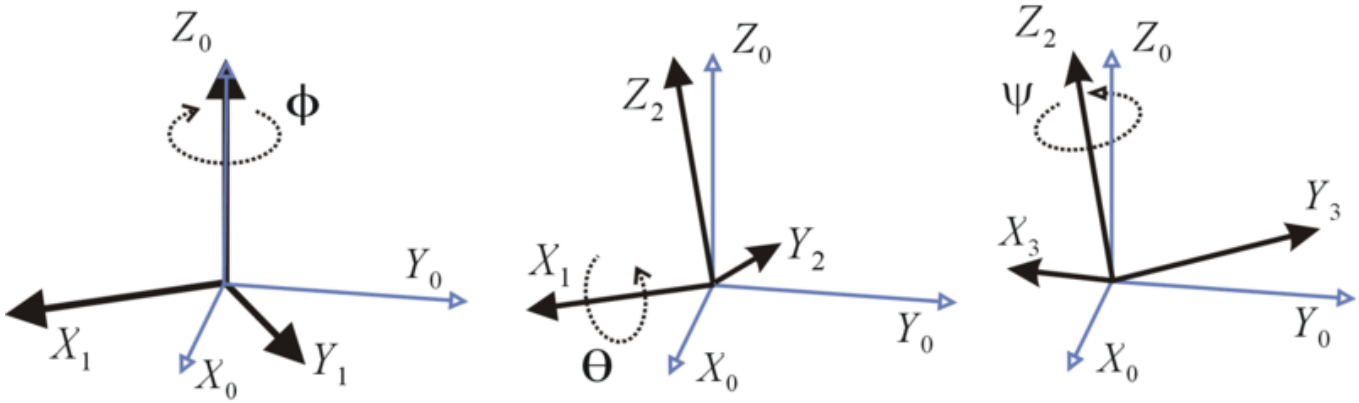
$$R(\alpha,\beta,\gamma) = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma\cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}$$

Euler Angles



$$R = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

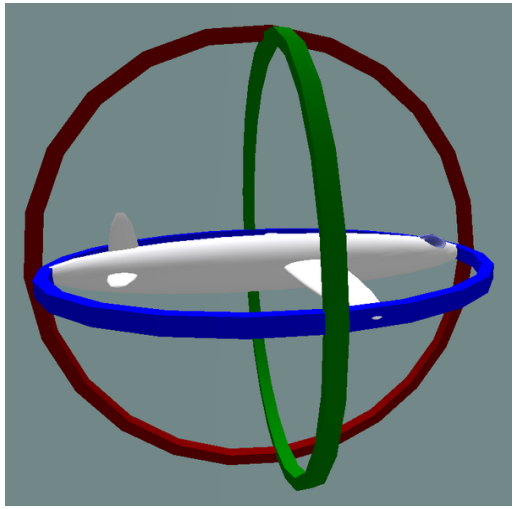
Euler Angles – Example: $-60°, 30°, 45°$

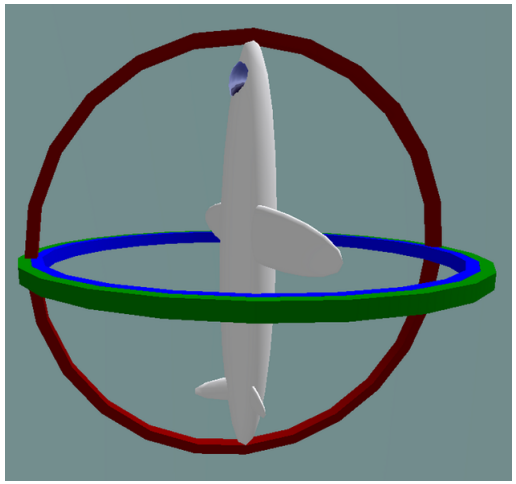


Problems With Representations

- Two major problems with yaw, pitch, roll and Euler angles:
- Cases where a continuum of values yield the same rotation matrix (no unique solution in certain cases).
- Cases where non-zero angles yield the identity rotation matrix which is equivalent to zero angles

Gimbal Lock



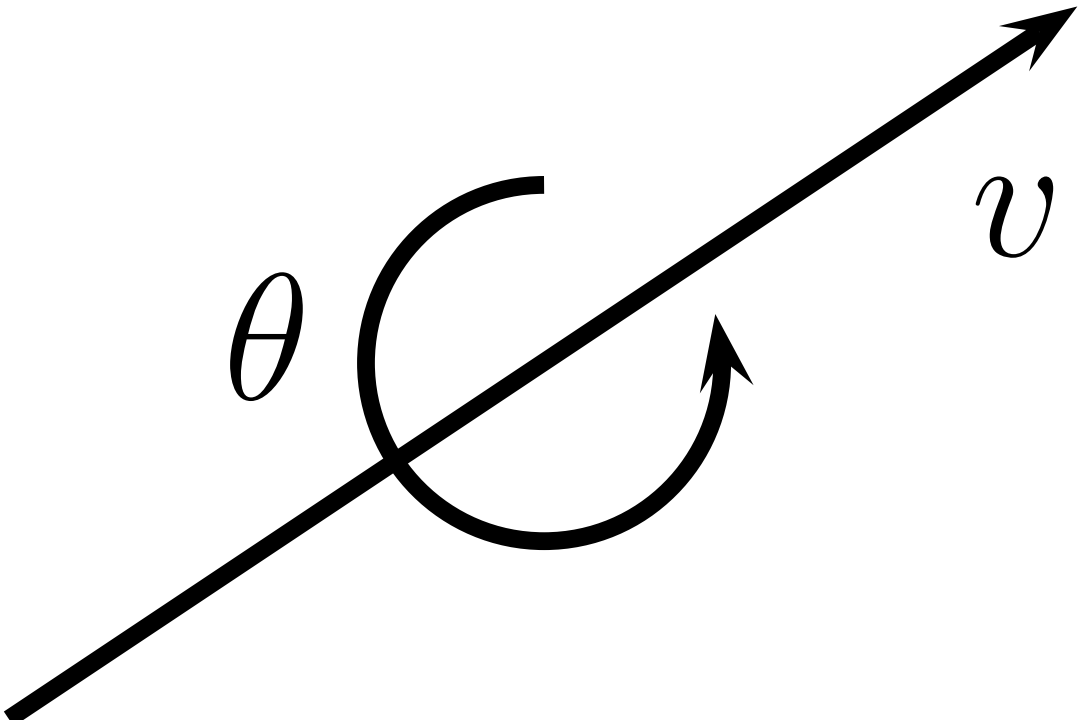


Example – when $\beta = 0$, Euler angle representation becomes:

$$R = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

Axis-Angle Representations



Axis-Angle Representations

- Given a rotation axis represented as a unit vector $\hat{k}$, rotating a vector v by an angle θ, a nice way to model the rotation is through *Rodrigues' formula*:

$$v_{rot} = v\cos\theta + (\hat{k} \times v)\sin\theta + \hat{k}(\hat{k} \cdot v)(1 - \cos\theta)$$

- **Explanation:** The plane of rotation is perpendicular to $\hat{k}$. Using the dot and cross products, the vector v can be decomposed into components parallel and perpendicular to the axis k as follows:

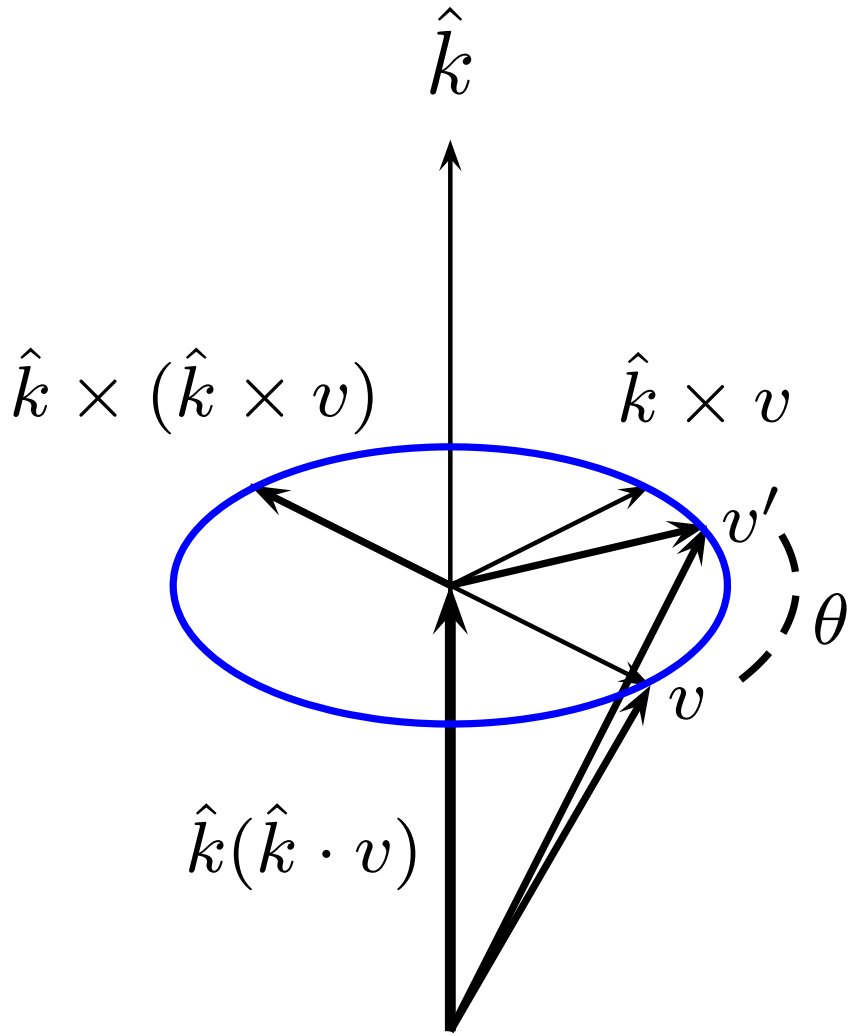
$$v = v_{\parallel} + v_{\perp}$$

  Where $v_{\parallel} = (v \cdot \hat{k})\hat{k}$ is the vector projection of v on the rotation axis $\hat{k}$.

- This part is parallel to the rotation axis and hence is not affected by the rotation.

Axis-Angle Representations

- The three vectors $\hat{k}$, $\hat{k} \times v$ and $\hat{k} \times (\hat{k} \times v)$ are three mutually perpendicular unit vectors
- $\hat{k} \times v$ is perpendicular to the plane defined by $\hat{k}$ and v, and $\hat{k} \times (\hat{k} \times v)$ is perpendicular to both $\hat{k}$ and $\hat{k} \times v$.
- Therefore $v_\perp = v - v_\parallel = v - (\hat{k} \cdot v)\hat{k} = -\hat{k} \times (\hat{k} \times v)$.



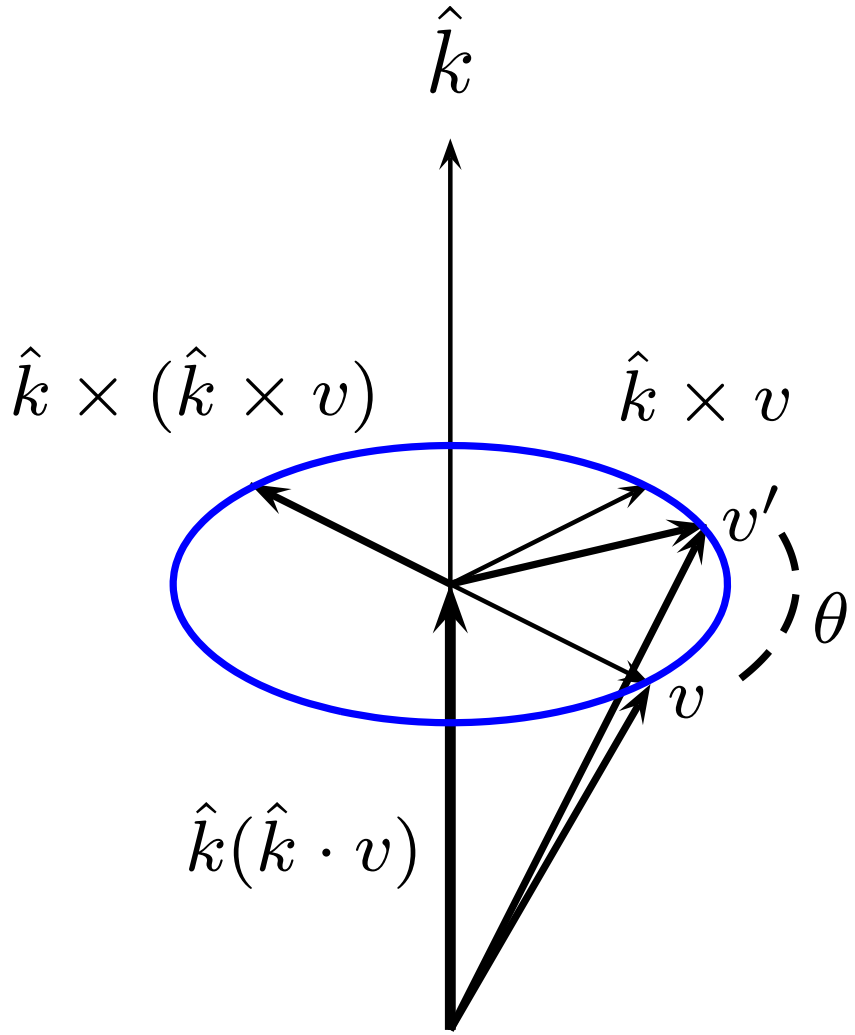
Axis-Angle Representations

- The only part that rotates is $v_\perp = v - (\hat{k} \cdot v)\hat{k}$, and it is a 2D rotation by θ around the plane perpendicular to $\hat{k}$:

$$\begin{aligned} v'_\perp &= (\hat{k} \times v) \sin\theta \\ &\quad - (\hat{k} \times (\hat{k} \times v)) \cos\theta \\ &= (\hat{k} \times v) \sin\theta + v \cos\theta \\ &\quad - (\hat{k} \cdot v)\hat{k} \cos\theta \end{aligned}$$

- Add to it the $v_\parallel$ component that did not change and get:

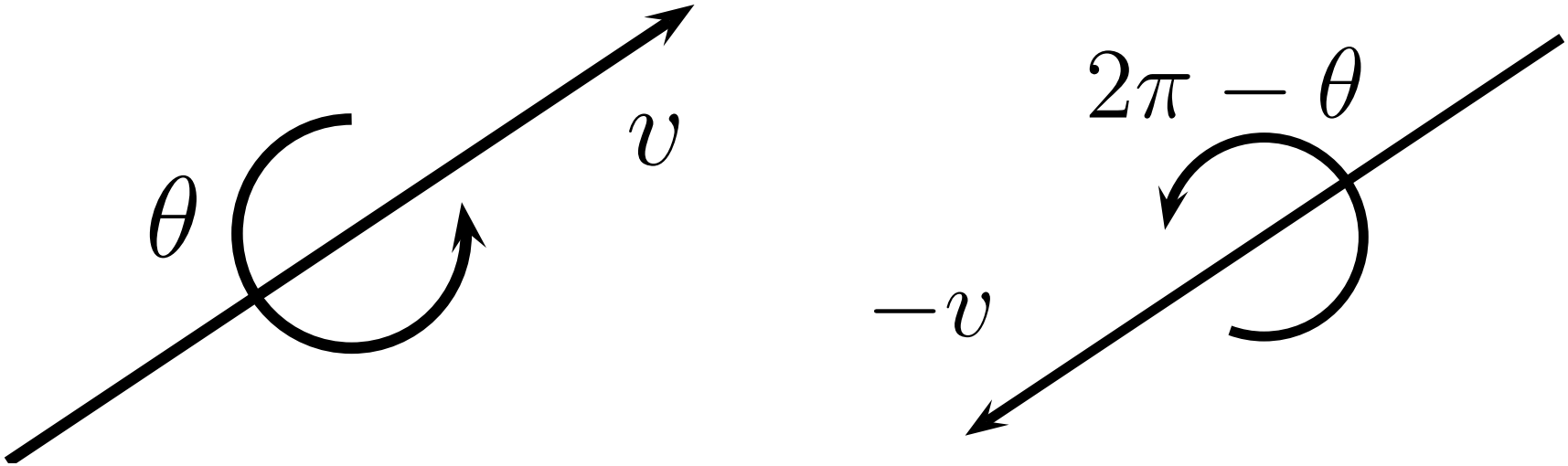
$$\begin{aligned} v' &= (\hat{k} \times v) \sin\theta + v \cos\theta \\ &\quad + (\hat{k} \cdot v)\hat{k}(1 - \cos\theta) \end{aligned}$$



Axis-Angle Representations Through Quaternions

- Quaternions are an extension of complex numbers.
- $h = a + bi + cj + dk$, a,b,c,d real numbers.
- i,j,k : imaginary components s.t.:
  - $i^2 = j^2 = k^2 = -1$
  - $ij = k$, $jk = i$, $ki = j$
  - $ij = -ji$, $jk = -kj$, $ki = -ik$
- Magnitude of a quaternion: $||h|| = \sqrt{a^2 + b^2 + c^2 + d^2}$
- a unit quaternion: $||h|| = 1$

Axis-Angle Representations Through Quaternions



$$h = \cos\frac{\theta}{2} + (v_x \sin\frac{\theta}{2})i + (v_y \sin\frac{\theta}{2})j + (v_z \sin\frac{\theta}{2})k$$

$$h = \cos\frac{\theta}{2} + v \sin\frac{\theta}{2}$$

$$-h = -\cos\frac{\theta}{2} - v \sin\frac{\theta}{2}$$

We assume that v is a unit vector!

Operations on Quaternions – Multiplication

- Given two quaternions – $h_1 = a_1 + ib_1 + jc_1 + kd_1$, $h_2 = a_2 + ib_2 + jc_2 + kd_2$
- Assume $v = [b, c, d]$, like a 3-D vector.
- $h1 \cdot h2 = (a_1 * a_2 - v_1 \cdot v_2, a_1 v_2 + a_2 v_1 + v_1 \times v_2)$
- $v_1 \cdot v_2$ is the dot product of v_1 and v_2, $v_1 \times v_2$ is the cross product.
- $h_1 \cdot h_2 = a_3 + ib_3 + jc_3 + kd_3$ Where:

$$
\begin{aligned}
a_3 &= a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2 \\
b_3 &= a_1 b_2 + a_2 b_1 + c_1 d_2 - c_2 d_1 \\
c_3 &= a_1 c_2 + a_2 c_1 + b_2 d_1 - b_1 d_2 \\
d_3 &= a_1 d_2 + a_2 d_1 + b_1 c_2 - b_2 c_1
\end{aligned}
$$

Operations on Quaternions – Rotation

- Given a unit quaternion $h = a + bi + cj + dk$, define its conjugate quaternion $h* = a - bi - cj - dk$:
- Transform point $p(x, y, z)$ by sandwiching: $h \cdot p \cdot h*$
- Treat p as a quaternion with no real component (a=0).
- The rotated point $p'(x', y', z')$ is obtained by the i,j,k components of the result
- To multiply a vector and a quaternion, see matrix representation above.
- Don't forget to translate the vector to the origin and translate back.

Operations on Quaternions – Combining Two Rotations

- Lemma: $(pq)^* = q^* p^*$.
- Sandwiching: $S_h(v) = h \cdot v \cdot h^*$
- $(S_{h_1} \circ S_{h_2})(v) = S_{h_1}(S_{h_2}(v)) = S_{h_1}(h_2 \cdot v \cdot h_2^*) = h_1(h_2 \cdot v \cdot h_2^*)h1^* = (h_1 h_2)v(h_2^* h_1^*) = S_{h_1 h_2}(v)$

Operations on Quaternions – Useful Examples

a	b	c	d	Description
1	0	0	0	Identity, no rotation
0	1	0	0	180° turn around X axis
0	0	1	0	180° turn around Y axis
0	0	0	1	180° turn around Z axis
$\sqrt{0.5}$	$\sqrt{0.5}$	0	0	90° rotation around X axis
$\sqrt{0.5}$	0	$\sqrt{0.5}$	0	90° rotation around Y axis
$\sqrt{0.5}$	0	0	$\sqrt{0.5}$	90° rotation around Z axis
$\sqrt{0.5}$	$-\sqrt{0.5}$	0	0	-90° rotation around X axis
$\sqrt{0.5}$	0	$-\sqrt{0.5}$	0	-90° rotation around Y axis
$\sqrt{0.5}$	0	0	$-\sqrt{0.5}$	-90° rotation around Z axis

from http://www.ogre3d.org/ .

Example – Rotation by 90° around Y axis

- $v = [0, 1, 0]$ (the rotation axis, which is the Y axis).
- $\theta = 90°$.
- $h = \cos\frac{\theta}{2} + (v_1 \sin\frac{\theta}{2})i + (v_2 \sin\frac{\theta}{2})j + (v_3 \sin\frac{\theta}{2})k =$
  $\sqrt{0.5} + 0 * i + \sqrt{0.5} * j + 0 * k$
- $h = \sqrt{0.5} + \sqrt{0.5} * j$
- $h* = \sqrt{0.5} - \sqrt{0.5} * j$
- Say $p = [1, 2, 3] = 1 * i + 2 * j + 3 * k$
- Transforming p:
  $p' = h \cdot p \cdot h* = (\sqrt{0.5}+\sqrt{0.5}*j) \cdot (i+2*j+3*k) \cdot (\sqrt{0.5}-\sqrt{0.5}*j)$

Example – Rotation by 90° around Y axis

- Transforming p:
  $p' = h \cdot p \cdot h* = (\sqrt{0}.5 + \sqrt{0}.5 * j) \cdot (i + 2 * j + 3 * k) \cdot (\sqrt{0}.5 - \sqrt{0}.5 * j)$
- $h_1 \cdot h_2 = (a_1 * a_2 - v_1 \cdot v_2, a_1 v_2 + a_2 v_1 + v_1 \times v_2)$
- $p \cdot h* = -[1, 2, 3] \cdot [0, -\sqrt{0.5}, 0], \sqrt{0}.5 * [1, 2, 3] + [1, 2, 3] \times [0, -\sqrt{0.5}, 0]...$
- $h \cdot p \cdot h* = 0, 3, 2, -1$

Quaternions Vs. Matrices

- A quaternion needs 4 doubles instead of 9
- Sandwiching takes 28 multiplications while matrices need 9
- Composing rotations takes 16 multiplications with quaternions and 27 for matrices
- When composing matrices, numerical inaccuracies lead to distortions. Vectors are no longer orthonormal and angles are distorted.
- Quaternions do not distort angles and renormalization is just a division by the quaternion magnitude : $q = q/|q|$
- In interpolation with matrices $R(t) = (1-t)R0 + tR1, R(t)$ does not represent a rotation.
- With $q(t) = (1-t)q0 + tq1, q(t)/|q(t)|$ is a valid rotation

Some Resources

- http://mathworld.wolfram.com/RotationMatrix.html
- http://mathworld.wolfram.com/EulerAngles.html
- http://mathworld.wolfram.com/Quaternion.html