Quaternions and Exponentials

Michael Kazhdan

(601.457/657)
Announcements

- Office hours:
  - Today at 1:30-2:50
  - Malone 229
Recall

We saw two different methods for interpolating/approximating between rotations:

- **Normalization**: (SVD) Blend as $3 \times 3$ matrices and then map to the closest rotation.
- **Parameterization**: (Euler Angles) Compute the parameter values, blend those, and then evaluate at the blended values.
Overview

• Cross Products and (Skew) Symmetric Matrices
• Quaternions
• The Exponential Map
Cross Product

Given 3D vectors $u = (u_1, u_2, u_3)$ and $v = (v_1, v_2, v_3)$ the cross product of $u$ and $v$ is:

$$u \times v = \begin{pmatrix}
  u_2 v_3 - u_3 v_2 \\
  u_3 v_1 - u_1 v_3 \\
  u_1 v_2 - u_2 v_1
\end{pmatrix}$$

Properties:

- The cross product is orthogonal to both $u$ and $v$.
- The vectors $u$, $v$, $u \times v$ align with the right hand rule.
- The length of the cross product is equal to the area of the parallelogram defined by $u$ and $v$.
- $u \times v = -v \times u$
- $u \times (v + w) = u \times v + u \times w$
- $(tu) \times v = t(u \times v)$
(Skew) Symmetric Matrices

A matrix $M$ is symmetric if:

$$M_{ij} = M_{ji} \iff M = M^t$$

A matrix $M$ is skew-symmetric if:

$$M_{ij} = -M_{ji} \iff M = -M^t$$

(Skew) Symmetric matrices are closed under addition and scaling:

- If $A = A^t$ and $B = B^t$, then $(A + B) = (A + B)^t$.
- If $A = A^t$, then $(\alpha A) = (\alpha A)^t$.
- If $A = -A^t$ and $B = -B^t$, then $(A + B) = -(A + B)^t$.
- If $A = -A^t$, then $(\alpha A) = -(\alpha A)^t$.  

Overview

• Cross Products and (Skew) Symmetric Matrices
• Quaternions
• The Exponential Map
Quaternions

Normalization:

- Find a representation of rotations that makes it easy to map the blend of rotations to the closest rotation
Quaternions

Quaternions are extensions of complex numbers, with three imaginary values instead of one:

\[ a + ib + jc + kd \]

Like the complex numbers, we can add quaternions together by summing the individual components:

\[
\begin{align*}
(a_1 + ib_1 + jc_1 + kd_1) \\
+ (a_2 + ib_2 + jc_1 + kd_2)
\end{align*}
\]

\[ = (a_1 + a_2) + i(b_1 + b_2) + j(c_1 + c_2) + k(d_1 + d_2) \]
Quaternions

Quaternions are extensions of complex numbers, with three imaginary values instead of one:
\[ a + ib + jc + kd \]

Like the imaginary component of complex numbers, squaring the components gives:
\[ i^2 = j^2 = k^2 = -1 \]

The multiplication rules are more complex:
\[ ij = k \quad ik = -j \quad jk = i \]
\[ ji = -k \quad ki = j \quad kj = -i \]
Quaternions

Quaternions are extensions of complex numbers, with three imaginary values instead of one:

$$a + ib + jc + kd$$

Like the imaginary component of complex numbers, squaring the components gives:

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

The multiplication rules are more complex:

$$ij = k \quad ik = -j \quad jk = i$$

$$ji = -k \quad ki = j \quad kj = -i$$

**Note:** Multiplication of quaternions is not commutative – the result of the multiplication depends on the order in which it is done.
Quaternions

More generally, the product of two quaternions is:

\[
(a_1 + ib_1 + jc_1 + kd_1) 
\times (a_2 + ib_2 + jc_2 + kd_2) 
= (a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2) 
+ i(a_1b_2 + a_2b_1 + c_1d_2 - c_2d_1) 
+ j(a_1c_2 + a_2c_1 - b_1d_2 + b_2d_1) 
+ k(a_1d_2 + a_2d_1 + b_1c_2 - b_2c_1)
\]

\[i^2 = j^2 = k^2 = -1\]
\[ij = k \quad ik = -j \quad jk = i\]
\[ji = -k \quad ki = j \quad kj = -i\]
Quaternions

As with complex numbers, the **conjugate** of a quaternion $q = a + ib + jc + kd$ is:

$$\bar{q} = a - ib - jc - kd$$

As with complex numbers, the **norm** of a quaternion $q = a + ib + jc + kd$ is:

$$\|q\| = \sqrt{a^2 + b^2 + c^2 + d^2} = \sqrt{q\bar{q}}$$

As with complex numbers, the **reciprocal** is defined by dividing the conjugate by the square norm:

$$\frac{1}{q} = \frac{1}{q} \cdot \frac{\bar{q}}{\bar{q}} = \frac{\bar{q}}{\|q\|^2}$$
**Quaternions**

One way to express a quaternion is as a pair consisting of the real value and the 3D vector consisting of the imaginary components:

\[ q = (\alpha, \vec{w}) \quad \text{with} \quad \alpha = a \quad \text{and} \quad \vec{w} = (b, c, d) \]

The advantage of this representation is that it is easier to express quaternion multiplication:

\[
q_1 \cdot q_2 = (\alpha_1, \vec{w}_1) \cdot (\alpha_2, \vec{w}_2) \\
= (\alpha_1 \cdot \alpha_2 - \langle \vec{w}_1, \vec{w}_2 \rangle, \alpha_1 \cdot \vec{w}_2 + \alpha_2 \cdot \vec{w}_1 + \vec{w}_1 \times \vec{w}_2)
\]

\[
q_1 \cdot q_2 = (a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2) \\
+ i(a_1 b_2 + a_2 b_1 + c_1 d_2 - c_2 d_1) \\
+ j(a_1 c_2 + a_2 c_1 - b_1 d_2 + b_2 d_1) \\
+ k(a_1 d_2 + a_2 d_1 + b_1 c_2 - b_2 c_1)
\]
Quaternions

We can also think of points in 3D as (purely imaginary) quaternions:

\[(x, y, z) \rightarrow ix + jy + kz = (0, \vec{w})\]

Given a unit quaternion \(q\) and an imaginary quaternion (3D point) \(p\), consider the map:

\[q(p) = qp\bar{q}\]
Quaternions

\[ q(p) = qp\bar{q} \]

\[ q_1 \cdot q_2 = (\alpha_1 \cdot \alpha_2 - \langle \bar{w}_1, \bar{w}_2 \rangle, \alpha_1 \cdot \bar{w}_2 + \alpha_2 \cdot \bar{w}_1 + \bar{w}_1 \times \bar{w}_2) \]

**Claim:**

1. The map takes 3D points to 3D points

\[ q(p) = (\alpha_q, \bar{w}_q)(0, \bar{w}_p)(\alpha_q, -\bar{w}_q) \]
\[ = (-\langle \bar{w}_q, \bar{w}_p \rangle, \alpha_q \bar{w}_p + \bar{w}_q \times \bar{w}_p)(\alpha_q, -\bar{w}_q) \]
\[ = (-\alpha_q \langle \bar{w}_q, \bar{w}_p \rangle + \alpha_q \langle \bar{w}_p, \bar{w}_q \rangle + \langle \bar{w}_q \times \bar{w}_p, \bar{w}_q \rangle, \ldots) \]
\[ = (0, \ldots) \]
Quaternions

\[
q(p) = qp\bar{q}
\]

\[
q_1 \cdot q_2 = (\alpha_1 \cdot \alpha_2 - \langle \vec{w}_1, \vec{w}_2 \rangle, \alpha_1 \cdot \vec{w}_2 + \alpha_2 \cdot \vec{w}_1 + \vec{w}_1 \times \vec{w}_2)
\]

Claim:

1. The map takes 3D points to 3D points
2. The map is linear

\[
q(a \cdot p_1 + b \cdot p_2) = q(a \cdot p_1 + b \cdot p_2)\bar{q}
\]

\[
= a \cdot qp_1\bar{q} + b \cdot qp_2\bar{q}
\]

\[
= a \cdot q(p_1) + b \cdot q(p_2)
\]
Quaternions

\[ q(p) = qp\bar{q} \]

\[ q_1 \cdot q_2 = (\alpha_1 \cdot \alpha_2 - \langle \vec{w}_1, \vec{w}_2 \rangle, \alpha_1 \cdot \vec{w}_2 + \alpha_2 \cdot \vec{w}_1 + \vec{w}_1 \times \vec{w}_2) \]

Claim:

1. The map takes 3D points to 3D points
2. The map is linear
3. The map is norm-preserving
   \[ |q(p)| = |qp\bar{q}| = |q||p||\bar{q}| = |p| \]
   since \( q \) is a unit quaternion
Quaternions

\[ q(p) = qp\bar{q} \]

\[ q_1 \cdot q_2 = (\alpha_1 \cdot \alpha_2 - \langle \vec{w}_1, \vec{w}_2 \rangle, \alpha_1 \cdot \vec{w}_2 + \alpha_2 \cdot \vec{w}_1 + \vec{w}_1 \times \vec{w}_2) \]

Claim:

1. The map takes 3D points to 3D points
2. The map is linear
3. The map is norm-preserving

\[ \Downarrow \]

The map \( p \rightarrow qp\bar{q} \) is a rotation when \( q \) is a unit quaternion.
Quaternions and Rotations

If $q = a + ib + jc + kd$ is a unit quaternion ($\|q\| = 1$), we can associate $q$ with the rotation:

$$R(q) = \begin{pmatrix}
1 - 2c^2 - 2d^2 & 2bc - 2ad & 2bd + 2ac \\
2bc + 2ad & 1 - 2b^2 - 2d^2 & 2cd - 2ab \\
2bd - 2ac & 2cd + 2ab & 1 - 2b^2 - 2c^2
\end{pmatrix}$$

Note that because all of the terms are quadratic, the rotation associated with $q$ is the same as the rotation associated with $-q$. 
Quaternions and Rotations

If \( q = a + ib + jc + kd \) is a unit quaternion (\( \|q\| = 1 \)), we can associate \( q \) with the rotation:

\[
R(q) = \begin{pmatrix}
1 - 2c^2 - 2d^2 & 2bc - 2ad & 2bd + 2ac \\
2bc + 2ad & 1 - 2b^2 - 2d^2 & 2cd - 2ab \\
2bd - 2ac & 2cd + 2ab & 1 - 2b^2 - 2c^2
\end{pmatrix}
\]

Because \( q \) is a unit quaternion, we have:

\[
\|q\|^2 = \|(a, \vec{w})\|^2 = a^2 + \|\vec{w}\|^2 = 1
\]

Or equivalently, if we set \( \hat{v} = \vec{w}/\|\vec{w}\| \), there exists \( \theta \) such that:

\[
q = \left( \cos \frac{\theta}{2}, \sin \frac{\theta}{2} \hat{v} \right)
\]
Quaternions and Rotations

If \( q = a + ib + jc + kd \) is a unit quaternion (\( \|q\| = 1 \)), we can associate \( q \) with the rotation:

\[
R(q) = \begin{pmatrix}
1 - 2c^2 - 2d^2 & 2bc - 2ad & 2bd + 2ac \\
2bc + 2ad & 1 - 2b^2 - 2d^2 & 2cd - 2ab \\
2bd - 2ac & 2cd + 2ab & 1 - 2b^2 - 2c^2
\end{pmatrix}
\]

Because \( q \) is a unit quaternion, we have:

\[
q = \left( \cos \frac{\theta}{2}, \sin \frac{\theta}{2} \vec{w} \right)
\]

It turns out that \( q \) corresponds to the rotation whose:

- axis of rotation is \( \vec{w} \), and
- angle of rotation is \( \theta \).
**Quaternions and Rotations**

If \( q = a + ib + jc + kd \) is a unit quaternion (\( ||q|| = 1 \)), we can associate \( q \) with the rotation:

\[
R(q) = \begin{pmatrix}
1 - 2c^2 - 2d^2 & 2bc - 2ad & 2bd + 2ac \\
2bc + 2ad & 1 - 2b^2 - 2d^2 & 2cd - 2ab \\
2bd - 2ac & 2cd + 2ab & 1 - 2b^2 - 2c^2
\end{pmatrix}
\]

Because \( q \) is a unit quaternion, we have:

In particular, if we express rotations in the axis-angle representation, we can compute the composition by multiplying quaternions.

It turns out that \( q \) corresponds to the rotation whose:

- axis of rotation is \( \vec{w} \), and
- angle of rotation is \( \theta \).
Quaternions

Instead of blending matrices \( \{M_0, \ldots, M_{n-1}\} \) and then normalizing using SVD, we can blend the quaternions and then normalize them:

- For each \( M_i \), compute the quaternion rep. \( (\alpha_i, \bar{w}_i) \)
Quaternions

Instead of blending matrices \( \{M_0, \ldots, M_{n-1}\} \) and then normalizing using SVD, we can blend the quaternions and then normalize them:

- For each \( M_i \), compute the quaternion rep. \((\alpha_i, \overline{\omega}_i)\)
- Interpolate/Approximate the quaternions:
Quaternions

Instead of blending matrices \( \{M_0, \ldots, M_{n-1}\} \) and then normalizing using SVD, we can blend the quaternions and then normalize them:

- For each \( M_i \), compute the quaternion rep. \((\alpha_i, \bar{w}_i)\)
- Interpolate/Approximate the quaternions:
  - Linear Interpolation:
    \[
    \alpha_i(t) = (1 - t)\alpha_i + t\alpha_{i+1} \\
    \bar{w}_i(t) = (1 - t)\bar{w}_i + t\bar{w}_{i+1}
    \]
Instead of blending matrices \( \{M_0, \ldots, M_{n-1}\} \) and then normalizing using SVD, we can blend the quaternions and then normalize them:

- For each \( M_i \), compute the quaternion rep. \((\alpha_i, \vec{w}_i)\)
- Interpolate/Approximate the quaternions:
  - Linear Interpolation
  - Catmull-Rom Interpolation:
    \[
    \alpha_i(t) = CR_0(t)\alpha_{k-1} + CR_1(t)\alpha_k + CR_2(t)\alpha_{k+1} + CR_3(t)\alpha_{k+2} \\
    \vec{w}_i(t) = CR_0(t)\vec{w}_{k-1} + CR_1(t)\vec{w}_k + CR_2(t)\vec{w}_{k+1} + CR_3(t)\vec{w}_{k+2}
    \]
Quaternions

Instead of blending matrices \( \{M_0, ..., M_{n-1}\} \) and then normalizing using SVD, we can blend the quaternions and then normalize them:

- For each \( M_i \), compute the quaternion rep. \((\alpha_i, \vec{w}_i)\)
- Interpolate/Approximate the quaternions:
  - Linear Interpolation
  - Catmull-Rom Interpolation
  - Uniform Cubic B-Spline Approximation:
    \[
    \alpha_i(t) = B_{0,3}(t)\alpha_{k-1} + B_{1,3}(t)\alpha_k + B_{2,3}(t)\alpha_{k+1} + B_{3,3}(t)\alpha_{k+2} \\
    \vec{w}_i(t) = B_{0,3}(t)\vec{w}_{k-1} + B_{1,3}(t)\vec{w}_k + B_{2,3}(t)\vec{w}_{k+1} + B_{3,3}(t)\vec{w}_{k+2}
    \]
Quaternions

Instead of blending matrices \( \{ M_0, ..., M_{n-1} \} \) and then normalizing using SVD, we can blend the quaternions and then normalize them:

- For each \( M_i \), compute the quaternion rep. \( (\alpha_i, \overrightarrow{w}_i) \)
- Interpolate/Approximate the quaternions:
  - Linear Interpolation
  - Catmull-Rom Interpolation
  - Uniform Cubic B-Spline Approximation
- Set the value of the in-between rotation to be the normalized quaternion:

\[
\Phi_i(t) = \frac{(\alpha_i(t), \overrightarrow{w}_i(t))}{\| (\alpha_i(t), \overrightarrow{w}_i(t)) \|}
\]
Quaternions

Instead of blending matrices \( \{M_0, \ldots, M_{n-1}\} \) and then normalizing using SVD, we can blend the quaternions and then normalize them:

- For each \( M_i \), compute the quaternion rep. \((\alpha_i, \vec{w}_i)\)
- Interpolate/Approximate the quaternions:
  - Linear Interpolation
  - Catmull-Rom Interpolation
  - Uniform Cubic B-Spline Approximation

Note:
- Using SVD, we interpolated in the \((9 = 3 \times 3)\)-dimensional space of matrices and then normalized.
- With quaternions we interpolate in the 4-dimensional space of quaternions and normalize.

\[
\Phi_i(t) = \frac{(\alpha_i(t), \vec{w}_i(t))}{\| (\alpha_i(t), \vec{w}_i(t)) \|}
\]
Quaternions

As with points on the circle/sphere, this type of interpolation/approximation has the limitation:

- Uniform sampling in quaternion space does not result in uniform sampling in rotation space.

\[
\Phi(t) = (1 - t)p_0 + tp_1
\]

\[
\tilde{\Phi}(t) = \frac{\Phi(t)}{\|\Phi(t)\|}
\]
Quaternions

As with points on the circle/sphere, this type of interpolation/approximation has the limitation:

- Uniform sampling in quaternion space does not result in uniform sampling in rotation space.

Additionally, since $R(-q) = R(q)$ there are two different quaternions we can associate with a rotation, so the mapping is not well-defined.
Overview

• Cross Products and (Skew) Symmetric Matrices
• Quaternions
• The Exponential Map
The Exponential Map

Parametrization:

• Find a canonical way to parametrize rotations so that there is little distortion
Geodesics

Given a surface $S(u, v)$ a geodesic is a curve that is (locally) the shortest path between two points.

$$S(u, v) = (\cos u \cos v, \sin u, \cos u \sin v)$$
Geodesics

Given a manifold (a $d$-dimensional surface) a geodesic is a curve that is (locally) the shortest path between two points.
Tangent Spaces

Given a curve $C(t)$, the tangent line to the curve at a point $p_0 = C(t_0)$ is the line that most closely approximates the curve $C(t)$ at the point $p_0$. 
Tangent Spaces

Given a curve $C(t)$, the tangent line to the curve at a point $p_0 = C(t_0)$ is the line that most closely approximates the curve $C(t)$ at the point $p_0$.

This is the line through $p_0$ with direction $C'(t_0)$.
Tangent Spaces

Given a surface $S(u, v)$ the tangent plane to the curve at a point $p_0 = S(u_0, v_0)$ is the plane that most closely approximates $S(u, v)$ at the point $p_0$.

$$S(u, v) = (\cos u \cos v, \sin u, \cos u \sin v)$$
Tangent Spaces

Given a surface $S(u, v)$ the tangent plane to the curve at a point $p_0 = S(u_0, v_0)$ is the plane that most closely approximates $S(u, v)$ at the point $p_0$. This is the plane through $p_0$, parallel to the plane spanned by:

$$\frac{\partial S(u,v)}{\partial u} \bigg|_{(u_0,v_0)} \quad \text{and} \quad \frac{\partial S(u,v)}{\partial v} \bigg|_{(u_0,v_0)}$$

$$S(u, v) = (\cos u \cos v, \sin u, \cos u \sin v)$$
Tangent Spaces

Given a manifold (a \(d\)-dimensional surface) the tangent space to the manifold at a point \(p_0\) on the manifold is the \(d\)-dimensional plane that most closely approximates the manifold at the point \(p_0\).
The Exponential Map

Given a curve $C(t)$, the exponential at $p_0 = C(t_0)$ is a map that sends points in the tangent space of $p_0$ to the curve $C(t)$. 
The Exponential Map

Given a curve $C(t)$, the exponential at $p_0 = C(t_0)$ is a map that sends points in the tangent space of $p_0$ to the curve $C(t)$.

The distance along the curve of a point $\exp_{p_0}(\vec{v})$ from $p_0$ is equal to $\|\vec{v}\|$.

Note: This is not the distance from the tangent line to the closest point on the curve.
The Exponential Map

Given a surface $S(u, v)$, the exponential at the point $p_0 = S(u_0, v_0)$ is a map that sends points in the tangent plane of $p_0$ to the surface $S(u, v)$. 

$\text{exp}(p_0, \bar{w})$
The Exponential Map

Given a surface $S(u, v)$, the exponential at the point $p_0 = S(u_0, v_0)$ is a map that sends points in the tangent plane of $p_0$ to the surface $S(u, v)$.

If we fix a vector $\vec{w}$ in the tangent space of $p_0$, then the curve $\exp_{p_0}(t\vec{w})$ will be a geodesic, leaving $p_0$ in direction $\vec{w}$ and will have length equal to $\|t\vec{w}\|$. 
The Exponential Map

Given a surface $S(u, v)$, the exponential at the point $p_0 = S(u_0, v_0)$ is a map that sends points in the tangent plane of $p_0$ to the surface $S(u, v)$.

If we fix a vector $\vec{w}$ in the tangent space of $p_0$, then the curve $\exp_{p_0}(t\vec{w})$ will be a geodesic, leaving $p_0$ in direction $\vec{w}$ and will have length equal to $\|t\vec{w}\|$.
The Exponential Map

Given a surface $S(u, v)$, the exponential at the point $p_0 = S(u_0, v_0)$ is a map that sends points in the tangent plane of $p_0$ to the surface $S(u, v)$.

If we fix a vector $\vec{w}$ in the tangent space of $p_0$, then the curve $\exp_{p_0}(t\vec{w})$ will be a geodesic, leaving $p_0$ in direction $\vec{w}$ and will have length equal to $\|t\vec{w}\|$.
The Exponential Map

Given a surface $S(u, v)$, the exponential at the point $p_0 = S(u_0, v_0)$ is a map that sends points in the tangent plane of $p_0$ to the surface $S(u, v)$.

If we fix a vector $\vec{w}$ in the tangent space of $p_0$, then the curve $\exp_{p_0}(t\vec{w})$ will be a geodesic, leaving $p_0$ in direction $\vec{w}$ and will have length equal to $\|t\vec{w}\|$. 
The Exponential Map

Given a surface $S(u, v)$, the exponential at the point $p_0 = S(u_0, v_0)$ is a map that sends points in the tangent plane of $p_0$ to the surface $S(u, v)$.

If we fix a vector $\vec{w}$ in the tangent space of $p_0$, then the curve $\exp_{p_0}(t\vec{w})$ will be a geodesic, leaving $p_0$ in direction $\vec{w}$ and will have length equal to $\|t\vec{w}\|$. 
The Exponential Map

Given a surface $S(u, v)$, the exponential at the point $p_0 = S(u_0, v_0)$ is a map that sends points in the tangent plane of $p_0$ to the surface $S(u, v)$.

If we fix a vector $\vec{w}$ in the tangent space of $p_0$, then the curve $\exp_{p_0}(t\vec{w})$ will be a geodesic, leaving $p_0$ in direction $\vec{w}$ and will have length equal to $\|t\vec{w}\|$.
The Exponential Map

Given a manifold (a \(d\)-dimensional surface), the exponential at point \(p_0\) on the manifold is a map that sends points in the tangent plane of \(p_0\) to the manifold.

If we fix a vector \(\vec{w}\) in the tangent space of \(p_0\), then the curve \(\exp_{p_0}(t\vec{w})\) will be a geodesic, leaving \(p_0\) in direction \(\vec{w}\) and will have length equal to \(\|t\vec{w}\|\).
The Logarithm Map

For a point $p_0$ on a curve/surface/manifold, the logarithm is the inverse of the exponential, sending points on the curve/surface/manifold back into the tangent space of $p_0$. 
The Exponential Map

Example:

Let $C$ be the unit circle, the exponential map $\exp_{(1,0)}(t)$ is the map sending the point $t$ to the point $(\cos t, \sin t)$.
The Exponential Map

**Example:**

Let $C$ be the unit circle, the exponential map $\exp_{(1,0)}(t)$ is the map sending the point $t$ to the point $(\cos t, \sin t)$.

**Note:**

The exponential map is many-to-one: $\exp_{(1,0)}(t) = \exp_{(1,0)}(t + 2k\pi)$ so the logarithm is not unique.
The Exponential Map

**Fact:**

- Let $GL(n)$ be the space of $n \times n$ invertible matrices, $\exp_{id}(M)$ is the map sending an arbitrary matrix $M$ to an invertible one.

- Let $SO(n)$ be the space of $n \times n$ rotation matrices, $\exp_{id}(M)$ is the map sending a skew-symmetric matrix $M$ to a rotation.
The Exponential Map

How do we actually compute the exponential map?
The Exponential Map

How do we actually compute the exponential map?

It is difficult to find a closed form solution, but for matrices we can use a Taylor series approximation:

$$\exp_{\text{id.}}(A) = \text{id.} + A + \frac{1}{2!} A^2 + \cdots + \frac{1}{n!} A^n + \cdots$$

In a similar manner, we can define the logarithm:

$$\ln_{\text{id.}}(A) = (A - \text{id.}) - \frac{(A-\text{id.})^2}{2} + \cdots + (-1)^{n+1} \frac{(A-\text{id.})^n}{n} + \cdots$$
The Exponential Map

Properties:

- $\exp_{\text{id.}}(0) = \text{id.}$

- $\left. \frac{\partial \exp_{\text{id.}}(tA)}{\partial t} \right|_{t=0} = A$

- $\exp_{\text{id.}}(\ln_{\text{id.}} A) = A$

- $\exp_{\text{id.}}(A + B) = \exp_{\text{id.}}(A) \cdot \exp_{\text{id.}}(B)$ if and only if $AB = BA$

  $\Rightarrow \exp_{\text{id.}}(-A) = (\exp_{\text{id.}}(A))^{-1}$

  $\Rightarrow \exp_{\text{id.}}(\alpha A) = (\exp_{\text{id.}}(A))^{\alpha}$
Rotation Interpolation/Approximation

Given a collection of rotations \( \{M_0, \ldots, M_{n-1}\} \) we can generate a curve passing through/near the matrices in the following manner:

- For each \( M_i \), compute the logarithm \( \ln_{id}(M_i) \)
Rotation Interpolation/Approximation

Given a collection of rotations \( \{M_0, \ldots, M_{n-1}\} \) we can generate a curve passing through/near the matrices in the following manner:

- For each \( M_i \), compute the logarithm \( \ln_{\text{id}}(M_i) \)
- Interpolate/Approximate the logarithms:
Rotation Interpolation/Approximation

Given a collection of rotations \( \{M_0, \ldots, M_{n-1}\} \) we can generate a curve passing through/near the matrices in the following manner:

- For each \( M_i \), compute the logarithm \( \ln_{\text{id}}(M_i) \)
- Interpolate/Approximate the logarithms:
  - Linear Interpolation:
    \[
    LN_k(t) = (1 - t) \ln_{\text{id}}(M_i) + \ln_{\text{id}}(M_{k+1})
    \]
Rotation Interpolation/Approximation

Given a collection of rotations \( \{M_0, ..., M_{n-1}\} \) we can generate a curve passing through/near the matrices in the following manner:

- For each \( M_i \), compute the logarithm \( \ln_{id}(M_i) \)
- Interpolate/Approximate the logarithms:
  - Linear Interpolation:
  - Catmull-Rom Interpolation:

\[
LN_k(t) = CR_0(t)\ln_{id}(M_{k-1}) + CR_1(t)\ln_{id}(M_k) + CR_1(t)\ln_{id}(M_{k+1}) + CR_1(t)\ln_{id}(M_{k+2})
\]
Rotation Interpolation/Approximation

Given a collection of rotations \( \{M_0, \ldots, M_{n-1}\} \) we can generate a curve passing through/near the matrices in the following manner:

- For each \( M_i \), compute the logarithm \( \ln_{\text{id.}}(M_i) \)
- Interpolate/Approximate the logarithms:
  - Linear Interpolation:
  - Catmull-Rom Interpolation:
  - Uniform Cubic B-Spline Approximation:

\[
LN_k(t) = B_{0,3}(t) \ln_{\text{id.}}(M_{k-1}) + B_{1,3}(t) \ln_{\text{id.}}(M_k) + B_{2,3}(t) \ln_{\text{id.}}(M_{k+1}) + B_{3,3}(t) \ln_{\text{id.}}(M_{k+2})
\]
Rotation Interpolation/Approximation

Given a collection of rotations \( \{M_0, \ldots, M_{n-1}\} \) we can generate a curve passing through/near the matrices in the following manner:

- For each \( M_i \), compute the logarithm \( \ln_{\text{id}}(M_i) \)
- Interpolate/Approximate the logarithms:
  - Linear Interpolation:
  - Catmull-Rom Interpolation:
  - Uniform Cubic B-Spline Approximation:
- Set the value of the in-between rotation to be the exponent of the blended logarithms:
  \[
  \Phi_k(t) = \exp_{\text{id}}(LN_k(t))
  \]
Rotation Interpolation/Approximation

Given a collection of rotations \( \{M_0, \ldots, M_{n-1}\} \) we can generate a curve passing through/near the matrices in the following manner:

- For each \( M_i \), compute the logarithm \( \ln_{\text{id}}(M_i) \)

Note:
Since the logarithm of rotations is a skew-symmetric matrix, and since skew-symmetric matrices are closed under addition and scaling, the weighted average \( LN_i(t) \) is also skew-symmetric, so its exponent has to be a rotation.

exponent of the blended logarithms:
\[
\Phi_k(t) = \exp_{\text{id}}(LN_k(t))
\]
Summary

In order to define in-between frames for an animation, we need to interpolate/approximate the transformations specified in the key-frames.

- For translation, we can just use splines.
- For rotations, we need to ensure that the in-between transformations are also rotations:
  - Euler angles
  - Exponential map \(\text{In-between transformations are guaranteed to be rotations}\)
  - SVD
  - Quaternions \(\text{Normalize in-between transformations to turn them into the nearest rotations}\)