Motion Interpolation & Sensor Fusion

Final presentation

Lennart Bastian
Antoine Keller
Sofia Morales Santiago

TUM

07.07.2018
Introduction

- Track 3D Objects with
  - Optical Tracking Systems, e.g. cameras
  - embedded system or inertial measurement units (IMU), e.g. accelerometers
Introduction

- Motion Interpolation
Introduction

- Motion Interpolation
  - Smoothness
Introduction

- Motion Interpolation
  - Smoothness
  - Accuracy

Sensor Fusion

OTS (e.g. cameras)
IMU (e.g. accelerometers)

⇒ combine to make the best motion interpolation
Introduction

- Motion Interpolation
  - Smoothness
  - Accuracy

**Sensor Fusion**

<table>
<thead>
<tr>
<th>OTS (e.g. cameras)</th>
<th>IMU (e.g. accelerometers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good tracking of translation</td>
<td>Almost impossible to track translation</td>
</tr>
<tr>
<td>Bad tracking of rotation</td>
<td>Good tracking of rotation</td>
</tr>
</tbody>
</table>
Introduction

- **Motion Interpolation**
  - Smoothness
  - Accuracy

- **Sensor Fusion**

<table>
<thead>
<tr>
<th>OTS (e.g. cameras)</th>
<th>IMU (e.g. accelerometers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good tracking of translation</td>
<td>Almost impossible to track translation</td>
</tr>
<tr>
<td>Bad tracking of rotation</td>
<td>Good tracking of rotation</td>
</tr>
</tbody>
</table>
**Introduction**

- **Motion Interpolation**
  - Smoothness
  - Accuracy

- **Sensor Fusion**

<table>
<thead>
<tr>
<th>OTS (e.g. cameras)</th>
<th>IMU (e.g. accelerometers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good tracking of translation</td>
<td>Almost impossible to track translation</td>
</tr>
<tr>
<td>Bad tracking of rotation</td>
<td>Good tracking of rotation</td>
</tr>
</tbody>
</table>

⇒ combine to make the best motion interpolation
Describing Rotations

Quaternions

- General Form

\[ q_1 1 + q_2 i + q_3 j + q_4 k = (q_1, q_2, q_3, q_4)^T \]

\[ i^2 = j^2 = k^2 = ijk = -1 \]

- Only represent rotations.
- Coordinate system independency.
- No gimbal lock.
- Simple interpolation.
Describing Rotations

Unit Quaternions

- Set $H_1$: $\| q \| = 1$.
- $H_1$ constitutes a hypersphere in quaternion space.
- The set is closed under multiplication.

Rotations

- $SO(3)$: space of three-dimensional rotations.
- Rotation about the axis $\nu = (\nu_1, \nu_2, \nu_3) \in \mathbb{R}^3$, angle $\theta$

$$q = [\cos(\theta/2), \sin(\theta/2)\nu]$$

- For each rotation there are 2 unit quaternions: $q$ and $-q$ (antipodal). $H_1$ is a "double-covering" of $SO_3$. 
Visualizing Quaternions

Explain here the process to convert quaternions trajectory on the unit sphere!
Basic Interpolation Methods

Let $q_0, q_1, \overline{q_0} \in \mathbb{H}$ and $h \in [0, 1]$:

- Liner Quaternion interpolation
  \[
  \text{Lerp}(q_0, q_1, h) = q_0(1 - h) + q_1 h.
  \]

- Spherical Linear Quaternion interpolation
  \[
  \text{SLERP}(q_0, q_1, h) = q_0(\overline{q_0 q_1})^h
  \]
Describing Rotations

**Rotation Axis of Slerp interpolation**

**Rotation Angle of Slerp Interpolation**

- **Keyframe**
  - 0
  - 100
  - 200
  - 300
  - 400

- **Angle ($\theta$) of rotation**
  - 0
  - 0.2
  - 0.4
  - 0.6
  - 0.8
Describing Rotations

(Slerp) Quaternion interpolation with rotation over the $x$ – axis
Linear Interpolation

- Most simple interpolation is piecewise linear.
- Given a sequence of points $p_i \in \mathbb{R}^n$, we can represent linear interpolation in a cumulative form:
Most simple interpolation is piecewise linear.

Given a sequence of points \( p_i \in \mathbb{R}^n \), we can represent linear interpolation in a cumulative form:

\[
p(t) = p_0 + \alpha_1(t)\Delta p_1 + \ldots + \alpha_n(t)\Delta p_n
\]

\[
= p_0 + \sum_{i=1}^{n} \alpha_i(t)\Delta p_i
\]

\( \alpha_i \) ramps from 0 to 1 in the interval \( i \leq t < i + 1 \).
Linear Interpolation

- Most simple interpolation is piecewise linear.
- Given a sequence of points $p_i \in \mathbb{R}^n$, we can represent linear interpolation in a cumulative form:

$$
p(t) = p_0 + \alpha_1(t)\Delta p_1 + ... + \alpha_n(t)\Delta p_n
$$

$$
= p_0 + \sum_{i=1}^{n} \alpha_i(t)\Delta p_i
$$

- $\alpha_i$ ramps from 0 to 1 in the interval $i \leq t < i + 1$.
- Similarly, we can construct a piece-wise quaternion slerp in a cumulative form:

$$
q(t) = q_0 \prod_{i=1}^{n} \omega_i^{\alpha_i(t)}
$$

with $\omega_i = \log(q_{i-1}^{-1}q_i)$, $q^t := \exp(t\log(q))$. 

Several different interpolation schemes can be represented in the cumulative form. $\alpha_i(t)$ is replaced with some basis function $\beta_i(t)$.

\[
q(t) = q_0 \prod_{i=1}^{n} \exp(\omega_i \beta_i(t))
\]

- B-splines
- Bezier curves
Several different interpolation schemes can be represented in the cumulative form. \( \alpha_i(t) \) is replaced with some basis function \( \beta_i(t) \).

\[
q(t) = q_0 \prod_{i=1}^{n} \exp(\omega_i \beta_i(t))
\]

- B-splines
- Bezier curves
- The cumulative form describes an **approximation**. In order to interpolate exactly a non-linear system of equations needs to be solved.
Bezier Curves

Figure 1: Bezier Curves and their Basis Functions

- global control / very smooth ($\in \mathbb{C}^{n-1}$)
- simple implementation

$^2$Anonymous Preprint 2018 [under review]
Bezier Curves

(a) Rotation Axis

(b) Rotation Angle

Figure 2: Rotation Axis and Angle of Bezier Approximation
Beziers Curves

Figure 3: Bezier Curve Interpolation on an Object
Optimization

- Smoothness
Optimization

- Smoothness
- Accuracy
Optimization
Goal: evaluate the distance between the initial quaternions \((q_{\text{orig}})_{i=1..N}\) and the interpolation path \((q_{\text{calc}}(p, t))_{i=1..N}\)

- Euclidean distance: \(\|q_{\text{orig},i} - q_{\text{calc},i}\|_2^2\)
- Angular distance: \(\text{Arccos}(2q_{\text{orig},i} \circ q_{\text{calc},i} - 1)\)
Optimization

- Smoothness
- Accuracy

\[
\min_p \sum \text{dist}(q(p, t_i) - q_i) + \lambda \int \|\ddot{q}(p, t)\|^2 dt
\]

- \(q_i\): original quaternions
- \(t_i\): times at which the calculated trajectory must fit the original quaternions
- \(p\): control points (our variable)
- \(q(p, t)\): calculated trajectory at time \(t\).
Animation Bezier Curve
Animation Bezier Curve
Trade-off between smoothness and accuracy

Figure 4: Bezier optimization for different $\lambda$

(a) $\lambda = 20$  
(b) $\lambda = 200$  
(c) $\lambda = 2000$
Result of Optimization

Figure 5: Optimized Bezier Interpolation
Solve the problem

\[
\min_p \sum \text{dist}(q(p, t_i) - q_i) + \lambda \int \|\ddot{q}(p, t)\|^2 dt
\]

- Compute the gradient
- Problem highly non-linear, not convex \(\Rightarrow\) Poor convergence
- Use Quasi-Newton method algorithm by providing the gradient

**Conclusion**: Use Matlab optimization function `fminunc`.
Performance : Running time

- Estimated gradient (numerical) method $\Rightarrow$ not as fast as an optimization with provided gradient.
- Order of magnitude : couple of minutes for 4 or 5 quaternions $\Rightarrow$ not made for real-time data. But the running time grows exponentially if you increase the number of quaternions.
Pose Estimation

Figure 6: ArUco Marker and IMU
IMU and OTS data is noise

Figure 7: OTS and IMU poses
Undesirable Minima

Figure 8: Undesirable minimum found through anti-podal point
Undesirable Minima

Figure 9: Rotation Angle of undesirable minimum at anti-podal point
Updated Optimization Model

- $J$ The set of captured OTS poses
- $I$ The set of captured IMU poses
- Quaternions are constrained to Northern Hemisphere

$$\min_p \quad \lambda_1 \sum_{i \in I} \| q(p, t_i) - q_i \| + \lambda_2 \sum_{j \in J} \| q(p, t_j) - q_j \| + \int \| \ddot{q}(p, t) \|^2 dt$$

s.t. $q_0 \geq 0$
Interpolation of OTS and IMU data
Interpolation of OTS and IMU data

Figure 10: Interpolation Rotation Axis
Conclusion

- Implemented Bezier curve interpolation for quaternions
- Designed an Optimization Model to improve interpolation
- Applied it to Sensor Fusion data provided by Framos
Thank you for your attention!

Figure 11: Optimized Bezier Interpolation
Sharp Turn

Figure 12: Rotation Axis and Angle of Sharp Twist

(a) \( \lambda = 3 \)  
(b) \( \lambda = 0.1 \)
Sharp Turn

Figure 13: Animation of a Sharp Turn