SKEL Force Visualization System

Joint Torque and GRF Visualization on SKEL Skeleton Mesh

Technical Documentation

January 11, 2026

Contents

1 Overview

This document describes the SKEL Force Visualization system, which visualizes joint torques and Ground Reaction Forces (GRF) from AddBiomechanics data on SKEL skeleton meshes. The visualization follows the PhysPT paper style with:

- **Skeleton mesh** colored by joint torque magnitude using the Plasma colormap
- **LBS-based bone segmentation** for proper per-bone coloring
- **3-axis torque lines** (X=Pink, Y=Green, Z=Cyan) with endpoint spheres
- **GRF arrows** (Red) from Center of Pressure in force direction
- **Body mesh** rendered separately in gray

2 Data Pipeline

2.1 Input Data Sources

1. **AddBiomechanics (.b3d files)**: Contains biomechanical data including:
   - Joint positions $\mathbf{p}_j \in \mathbb{R}^3$ for each joint j
   - Joint torques $\boldsymbol{\tau}_j \in \mathbb{R}^{n_j}$ where n_j is the DOF count
   - Ground reaction forces (GRF) and center of pressure (CoP)
2. **SKEL Mesh Files**: Generated from AddBiomechanics to SKEL conversion:
   - `skel_mesh.obj`: Body surface mesh
   - `skel_skeleton.obj`: Internal skeleton bone mesh (∼10MB, detailed bone geometry)
3. **Force Data JSON**: Preprocessed per-frame data:
   - Joint positions, torque magnitudes, and tau vectors
   - GRF vectors, CoP positions, and contact flags

3 Joint Mapping System

3.1 SKEL 24 Joints

The SKEL model uses 24 joints indexed as follows:

Index	Joint Name	Index	Joint Name
0	pelvis	12	thorax
1	femur_r	13	head
2	tibia_r	14	scapula_r
3	talus_r	15	humerus_r
4	calcn_r	16	ulna_r
5	toes_r	17	radius_r
6	femur_l	18	hand_r
7	tibia_l	19	scapula_l
8	talus_l	20	humerus_l
9	calcn_l	21	ulna_l
10	toes_l	22	radius_l
11	lumbar	23	hand_l

Table 1: SKEL 24 Joint Index Mapping

3.2 AddBiomechanics to SKEL Joint Mapping

AddBiomechanics joint names are mapped to SKEL joint indices. **Important:** Some AddBiomechanics joints map to multiple SKEL joints (one-to-many mapping):

AddBiomechanics Joint	SKEL Indices
ground_pelvis	[0] (pelvis)
hip_r	[1] (femur_r)
walker_knee_r	[2] (tibia_r)
ankle_r	[3] (talus_r)
subtalar_r	[4] (calcn_r)
mtp_r	[5] (toes_r)
hip_l	[6] (femur_l)
walker_knee_l	[7] (tibia_l)
ankle_l	[8] (talus_l)
subtalar_l	[9] (calcn_l)
mtp_l	[10] (toes_l)
back	**[11, 12, 13]** (lumbar + thorax + head)
acromial_r	**[14, 15]** (scapula_r + humerus_r)
elbow_r	[16] (ulna_r)
radioulnar_r	[17] (radius_r)
radius_hand_r	[18] (hand_r)
acromial_l	**[19, 20]** (scapula_l + humerus_l)
elbow_l	[21] (ulna_l)
radioulnar_l	[22] (radius_l)
radius_hand_l	[23] (hand_l)

Table 2: AddBiomechanics to SKEL Joint Mapping (one-to-many mappings in bold)

3.3 Parent Joint Fallback for Missing Torque Data

AddBiomechanics does not provide torque data for certain joints (e.g., `mtp_r`, `radioulnar_r`). The system inherits torque from parent joints using a recursive fallback chain:

Missing Joint	Fallback Parent	Description
mtp_r (toes)	subtalar_r	Toe inherits from ankle
mtp_l (toes)	subtalar_l	Toe inherits from ankle
radioulnar_r	elbow_r	Forearm rotation inherits from elbow
radioulnar_l	elbow_l	Forearm rotation inherits from elbow
radius_hand_r	elbow_r	Wrist inherits from elbow
radius_hand_l	elbow_l	Wrist inherits from elbow
subtalar_r	ankle_r	Heel inherits from ankle
subtalar_l	ankle_l	Heel inherits from ankle
ankle_r	walker_knee_r	Ankle inherits from knee (if missing)
ankle_l	walker_knee_l	Ankle inherits from knee (if missing)

Table 3: Parent Joint Fallback Chain

The fallback is applied recursively: if the parent joint also has no torque data, the system continues up the kinematic chain until valid data is found.

4 LBS-based Bone Segmentation

4.1 Linear Blend Skinning Weights

The SKEL template mesh has 247,252 vertices, each with skinning weights for 24 joints:

$$\mathbf{W} \in \mathbb{R}^{247252 \times 24}, \quad \sum_{j=1}^{24} W_{v,j} = 1 \quad \forall v \tag{1}$$

4.2 Dominant Joint Assignment

Each vertex is assigned to its dominant joint:

$$\text{dominant\_joint}(v) = \arg \max_{j \in \{0,\ldots,23\}} W_{v,j} \tag{2}$$

4.3 Vertex Count Mismatch Handling

Saved meshes may have deduplicated vertices (e.g., 115,010 vs 247,252). The system handles this via face-based vertex mapping:

```
Algorithm: Face-based Vertex Mapping
Input: Deduplicated vertices V_dedup, faces F
       Template vertices V_template, dominant joints D_template
Output: Dominant joints for deduplicated mesh D_dedup

For each vertex v_i in V_dedup:
    1. Find faces containing v_i
```

```
   2. Get original template face indices
   3. Map to template vertex index via face correspondence
   4. D_dedup[i] = D_template[template_idx]
```

4.4 SKEL Joint Vertex Distribution

Each joint controls a specific number of vertices:

Idx	Joint	Vertices	Idx	Joint	Vertices
0	pelvis	8,456	12	thorax	15,234
1	femur_r	3,421	13	head	22,156
2	tibia_r	2,876	14	scapula_r	4,123
3	talus_r	99	15	humerus_r	2,345
4	calcn_r	1,234	16	ulna_r	1,567
5	toes_r	1,595	17	radius_r	236
6-10	(left leg)	similar	18	hand_r	41,196
11	lumbar	5,678	19-23	(left arm)	similar

Table 4: Approximate vertex count per joint

5 Algorithms

5.1 Plasma Colormap

The Plasma colormap maps torque magnitude to color (purple → yellow):

```
Algorithm: Torque to Plasma Color
Input: Torque magnitude m, max_torque m_max (default: 300 Nm)
Output: RGB color tuple (r, g, b) in [0,1]^3

1. If m < 0.01: return (0.8, 0.8, 0.8)  # Light gray for no torque

2. t = log10(m + 1) / log10(m_max + 1)  # Log-scale normalization
3. t = clip(t, 0, 1)

4. Plasma control points:
   C0 = (0.050, 0.030, 0.528)  # t=0.0: Dark purple
   C1 = (0.418, 0.001, 0.658)  # t=0.25: Purple
   C2 = (0.798, 0.280, 0.470)  # t=0.5: Magenta
   C3 = (0.988, 0.558, 0.231)  # t=0.75: Orange
   C4 = (0.940, 0.975, 0.131)  # t=1.0: Yellow

5. i = floor(4*t), f = 4*t - i
6. If i >= 4: return C4
7. return (1-f)*C_i + f*C_{i+1}  # Linear interpolation
```

5.2 LBS-based Vertex Coloring

Each skeleton mesh vertex is colored based on its dominant joint's torque:

```
Algorithm: LBS-based Skeleton Vertex Coloring
Input: Skeleton vertices V, faces F
       LBS weights W, joint torques T
       AddB to SKEL mapping M, parent fallback P
Output: Vertex colors C

Step 1: Build SKEL index to torque mapping
  For each AddB joint name a with torque tau_a:
    For each SKEL index s in M[a]:
      T_skel[s] = max(T_skel[s], tau_a)

Step 2: Apply parent fallback
  For each (child, parent) in P:
    For each SKEL index s in M[child]:
      If T_skel[s] = 0:
        p = parent
        While T[p] = 0 and p in P:
          p = P[p]  # Recursive fallback
        If T[p] > 0:
          T_skel[s] = T[p]

Step 3: Assign colors
  For each vertex v_i in V:
    j = dominant_joint(v_i)
    C[i] = PlasmaColor(T_skel[j])
  Return C
```

5.3 3-Axis Torque Line Generation

For each joint, three axis-aligned lines represent the X, Y, Z components:

```
Algorithm: 3-Axis Torque Line Generation
Input: Joint position p in R^3
       Torque vector tau in R^n (1-6 DOFs)
       Scale factor s (default: 0.002 m/Nm)
       Threshold theta (default: 0.5 Nm)
Output: Line meshes L and endpoint spheres S

1. Pad tau to 3D:
   If n = 1: tau_3D = (0, tau_1, 0)  # 1-DOF: Y-axis rotation
   Elif n = 2: tau_3D = (tau_1, tau_2, 0)
   Else: tau_3D = (tau_1, tau_2, tau_3)

2. Axis colors:
   c_X = (1.0, 0.4, 0.7)  # Pink
   c_Y = (0.4, 1.0, 0.4)  # Neon Green
   c_Z = (0.4, 0.8, 1.0)  # Cyan

3. For axis a in {X, Y, Z}:
   m_a = |tau_3D[a]|
   If m_a < theta: continue
```

```
length = m_a * s
sign = sgn(tau_3D[a])
e = p + sign * d_a * length

L_a = CreateCylinder(p, e, r=0.004)
S_a = CreateSphere(e, r=0.012)
```

6 Ground Reaction Force (GRF) Visualization

6.1 GRF Arrow Parameters

Parameter	Default Value	Description
`show_grf`	True	Enable GRF visualization
`grf_scale`	0.001 m/N	Arrow length scale (710N $\rightarrow$ 71cm)
`grf_radius`	0.008 m	Arrow shaft radius
`grf_color`	(1.0, 0.2, 0.2)	Bright red

Table 5: GRF Visualization Parameters

6.2 GRF Arrow Generation Algorithm

```
Algorithm: GRF Arrow Generation
Input: GRF list from force_data.json
       Scale s, radius r, color c
Output: Arrow meshes (shaft + head)

For each GRF entry:
  If contact = 0: continue  # Skip non-contact

  CoP = center of pressure position
  F = force vector [F_x, F_y, F_z]
  |F| = force magnitude

  If |F| < 1 N: continue  # Skip negligible forces

  F_hat = F / |F|  # Normalize direction
  length = |F| * s  # Arrow length
  e = CoP + F_hat * length  # Arrow endpoint

  Create cylinder from CoP to e with radius r
  Create sphere at e with radius 2r (arrow head)
```

6.3 GRF Data Structure

Example GRF entry from `force_data.json`:

```
{
    "body_idx": 0,            # 0=right foot, 1=left foot
    "grf": [-102.5, 702.7, 21.3],   # Force vector [N]
    "cop": [-0.308, 0.0, 0.221],    # Center of Pressure [m]
```

```
    "magnitude": 710.41,      # |F| in Newtons
    "contact": 1              # 1=contact, 0=no contact
}
```

7 Output Format

7.1 OBJ with Vertex Colors

The output OBJ files use the extended vertex format with RGB colors:

```
# SKEL skeleton mesh + 3-axis torque lines + GRF arrows
# Skeleton: colored by joint torque magnitude
# Axes: X=Pink, Y=Neon Green, Z=Cyan (with endpoint balls)
# GRF: Red arrows from CoP in force direction

# Vertex format: v x y z r g b
v 0.123456 0.789012 0.345678 0.9400 0.9752 0.1313
v ...

# Face format (1-indexed)
f 1 2 3
f ...
```

7.2 Output Files per Frame

- `frame_XXXX_skeleton_axes.obj`: Combined skeleton mesh + 3-axis lines + GRF arrows
- `frame_XXXX_body.obj`: Body surface mesh (gray)
- `frame_XXXX_force.txt`: Human-readable torque summary

8 Python Module Structure

```
skel_force_vis/
|-- __init__.py          # Package initialization
|-- __main__.py          # CLI entry point
|-- colormaps.py         # Plasma colormap and axis colors
|-- mesh_utils.py        # Cylinder, sphere, OBJ I/O
|-- lbs_utils.py         # LBS weights, joint mapping, fallback
|-- visualizer.py        # SKELForceVisualizer class
|-- run.py               # Command-line interface
|-- documentation.tex    # This document
```

9 Usage Examples

9.1 Python API

```
from skel_force_vis import SKELForceVisualizer

vis = SKELForceVisualizer(
```

```
    input_base="/path/to/skel_force_vis/Falisse2017_subject_1",
    output_dir="/path/to/output",
    skel_model_path="/path/to/skel_models_v1.1",
    gender='male',
    use_lbs_coloring=True,
    max_torque=300.0,
    show_grf=True,
    grf_scale=0.001,
)

results = vis.process_all_frames()
# Output: frame_0050: 14 joints, 48 axes, 2 GRF
```

9.2 Configuration Parameters

Parameter	Description	Default Value
`max_torque`	Colormap normalization max	300 Nm
`line_scale`	Torque to length conversion	0.002 m/Nm
`line_radius`	Axis line cylinder radius	0.004 m
`sphere_radius`	Endpoint sphere radius	0.012 m
`torque_threshold`	Minimum torque to display	0.5 Nm
`use_lbs_coloring`	Use LBS weights for coloring	True
`show_grf`	Display GRF arrows	True
`grf_scale`	GRF arrow length scale	0.001 m/N
`grf_radius`	GRF arrow shaft radius	0.008 m

Table 6: SKELForceVisualizer Configuration Parameters

10 Color Scheme Summary

- **Skeleton Mesh (Plasma colormap)**:
  - Low torque (<10 Nm): Dark purple
  - Medium torque (∼50 Nm): Magenta
  - High torque (>200 Nm): Yellow
  - No torque data: Light gray
- **Torque Axis Lines**:
  - X-axis: Pink $(1.0, 0.4, 0.7)$
  - Y-axis: Neon Green $(0.4, 1.0, 0.4)$
  - Z-axis: Cyan $(0.4, 0.8, 1.0)$
- **GRF Arrows**: Bright Red $(1.0, 0.2, 0.2)$
- **Body Mesh**: Gray $(0.7, 0.7, 0.7)$

11 Example Output Analysis

11.1 Frame 50 - Falisse2017_subject_1

```
Joint Torques (sorted by magnitude):
  ground_pelvis       :   786.22 Nm  (Yellow - highest)
  hip_r               :   137.56 Nm  (Orange)
  back                :    74.57 Nm  (Orange/Magenta)
  hip_l               :    68.51 Nm  (Orange/Magenta)
  acromial_r          :    13.47 Nm  (Purple)
  acromial_l          :    11.12 Nm  (Purple)
  ankle_r             :     7.99 Nm  (Purple/Blue)
  walker_knee_l       :     5.04 Nm  (Blue)
  subtalar_r          :     4.08 Nm  (Blue)
  walker_knee_r       :     3.08 Nm  (Blue)
  elbow_r             :     2.57 Nm  (Blue)
  elbow_l             :     1.69 Nm  (Blue)
  subtalar_l          :     0.73 Nm  (Dark Blue)
  radius_hand_l       :     0.54 Nm  (Dark Blue)

GRF (Right foot contact):
  Magnitude: 710.41 N
  Direction: [-102.5, 702.7, 21.3] N (mostly vertical)
  CoP: [-0.308, 0.0, 0.221] m
```

Note: The foot appears blue despite GRF=710N because joint torque (ankle ≈8 Nm) is different from GRF. GRF is the external force from the ground; joint torque is the internal moment generated by muscles.

12 References

- **AddBiomechanics**: `https://addbiomechanics.org/`
- **SKEL**: Keller et al., "SKEL: A Parametric Body Model with Bones"
- **PhysPT**: Physics-aware pretrained transformer for human motion
- **Plasma Colormap**: Matplotlib perceptually uniform colormap