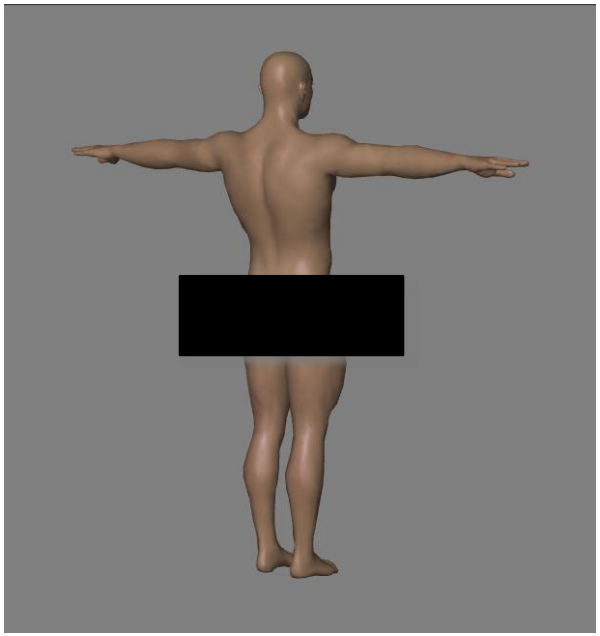
Skinning

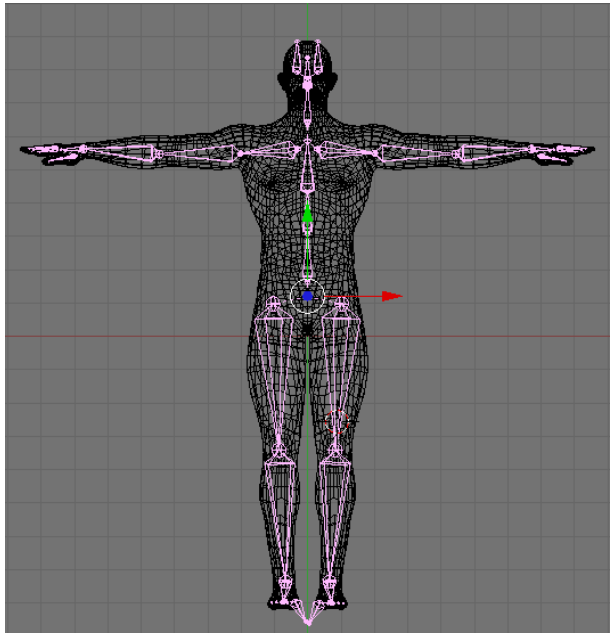
content

CHARACTER ANIMATION

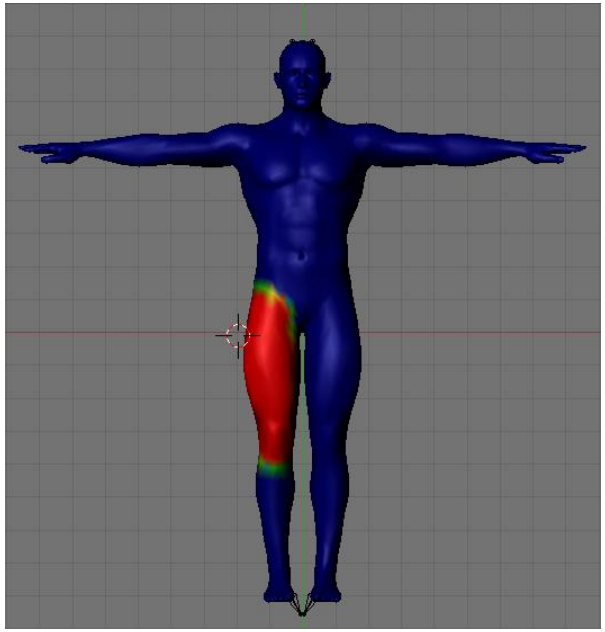
Skeletal Animation



design or scan mesh in reference pose



build skeleton, align with mesh and adapt bone lengths



for each bone describe influence on vertices by assigning weights

Overview

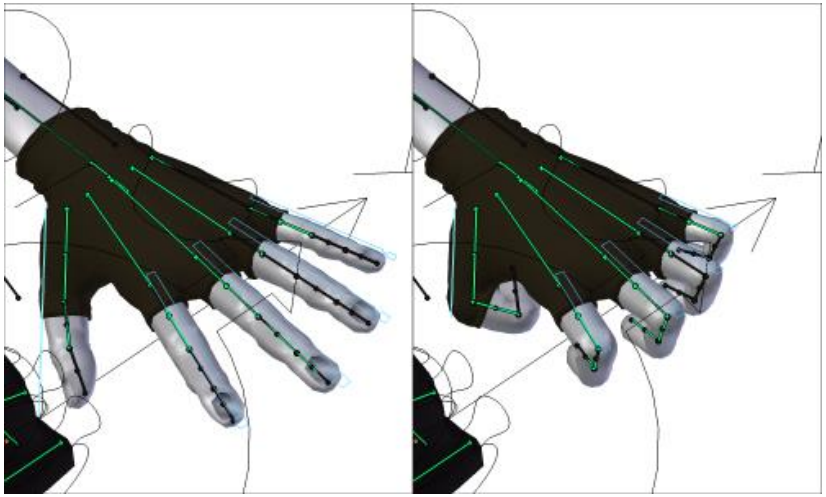
- Rigging: map skeleton to mesh in reference pose
- Animation: define new pose on skeleton
- Skinning: map pose from skeleton to mesh

Skeletal Animation

important applications

- body
- face
- hands

example for hand animation:



© wikipedia

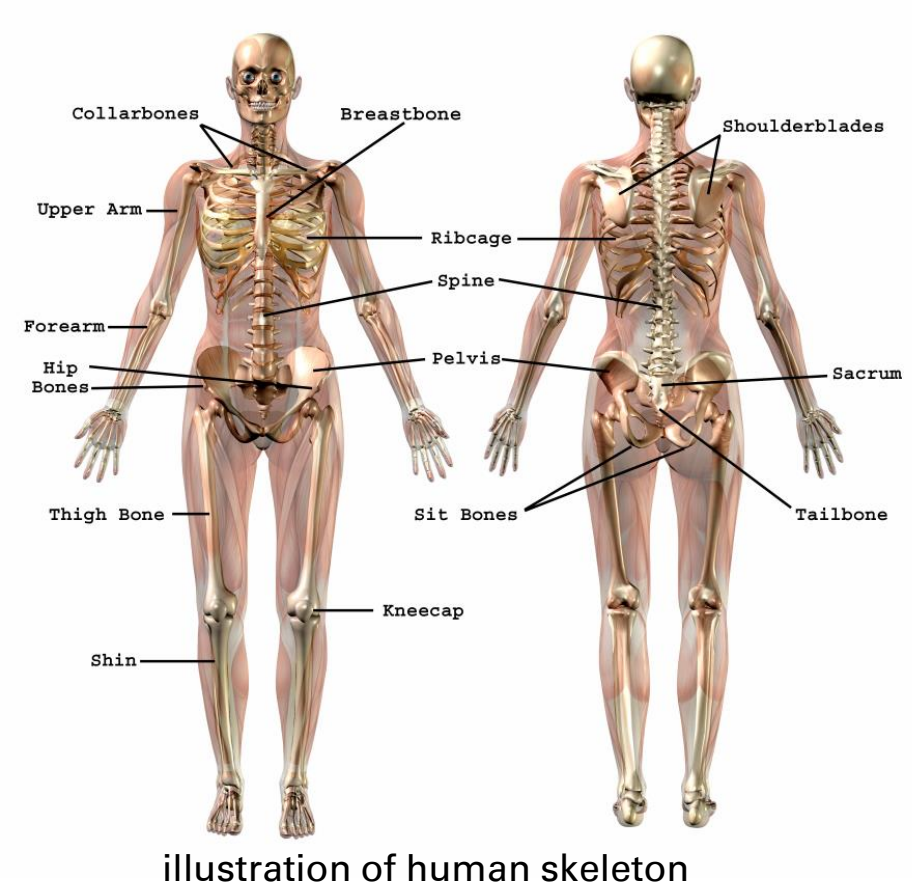
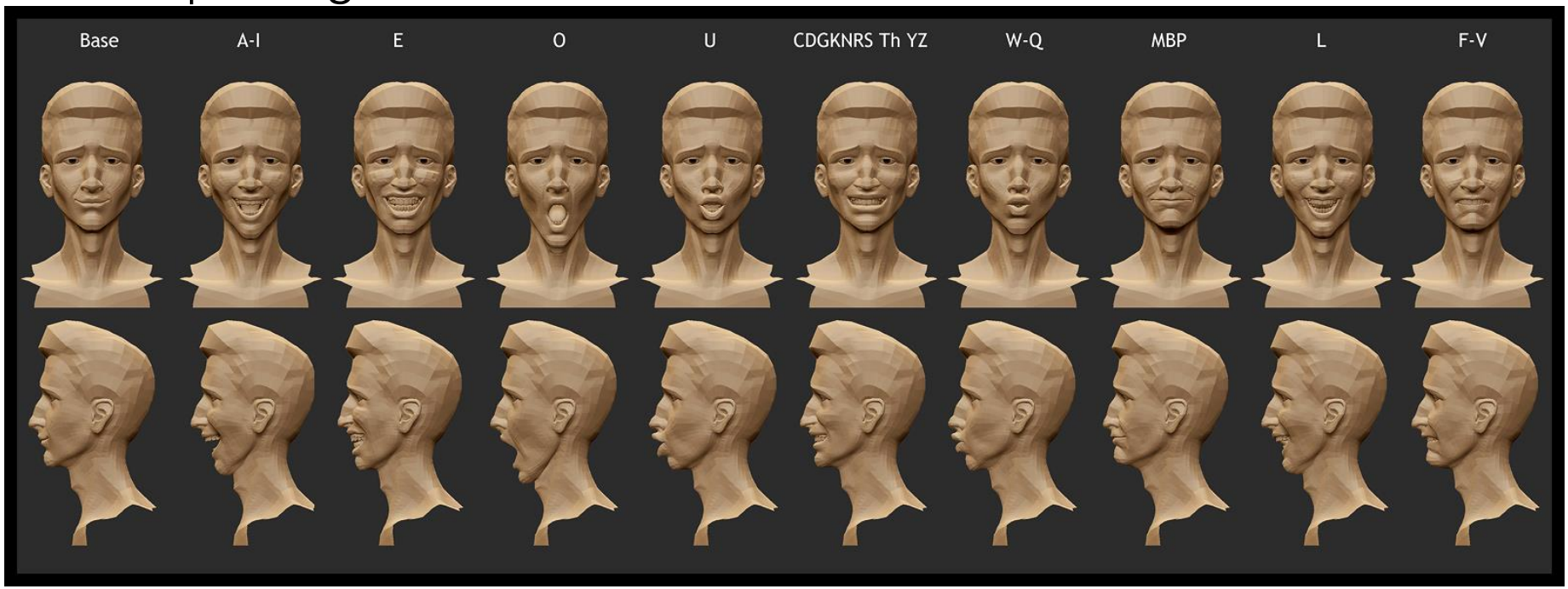


illustration of human skeleton

© http://insectanatomy.com/tag/bones-names

Morph Target Animation

- **design:** given a mesh, e.g. of a face, in a natural pose, model a large number of localized deformations to provide different expressions of the mesh, called morph targets
- **animation:** provide an animation parameter for each morph target to allow blending in an arbitrary number of morph targets



Physically-Based Approaches

- Main idea:
  - Simulate bones, muscles, tissue and skin
- Advantages
  - Very realistic
- Disadvantages
  - Computational cost
  - Large number of parameters to adapt for each animated individual

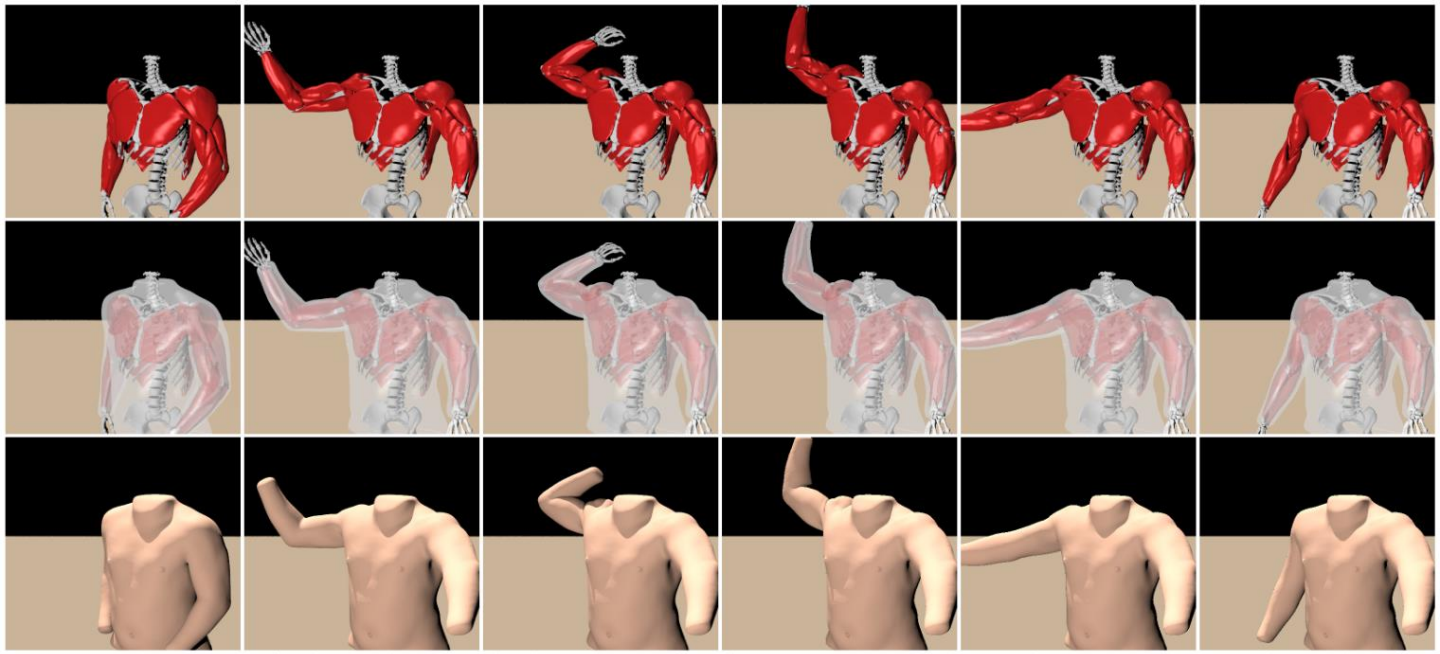


Figure 6: *Illustration of a layered approach where the results of a quasistatic muscle simulation are subsequently used to drive a quasistatic simulation of the outer flesh.*

image taken from Teran et al, Robust Quasistatic Finite Elements and Flesh Simulation, SCA 2005

Scanning, Examples

- Main idea
  - Capture skin deformation of real people
  - Learn skin deformation from these examples
- Advantages
  - Very realistic
- Disadvantages
  - Needs scanning devices
  - Need many examples
  - Memory consumption

T. Neumann[1], K. Varanasi[3], N. Hasler[3], M. Wacker[1], M. Magnor[2], C. Theobalt[3]

Capture and Statistical Modeling of Arm Muscle Deformation

Eurographics 2013

1 HTW Dresden, Germany
2 Computer Graphics Lab, TU Braunschweig, Germany
3 Max-Planck-Institut, Saarbrücken, Germany

www.youtube.com/watch?v=L4T4t2_qkDk

https://doi.org/10.1111/cgf.12048

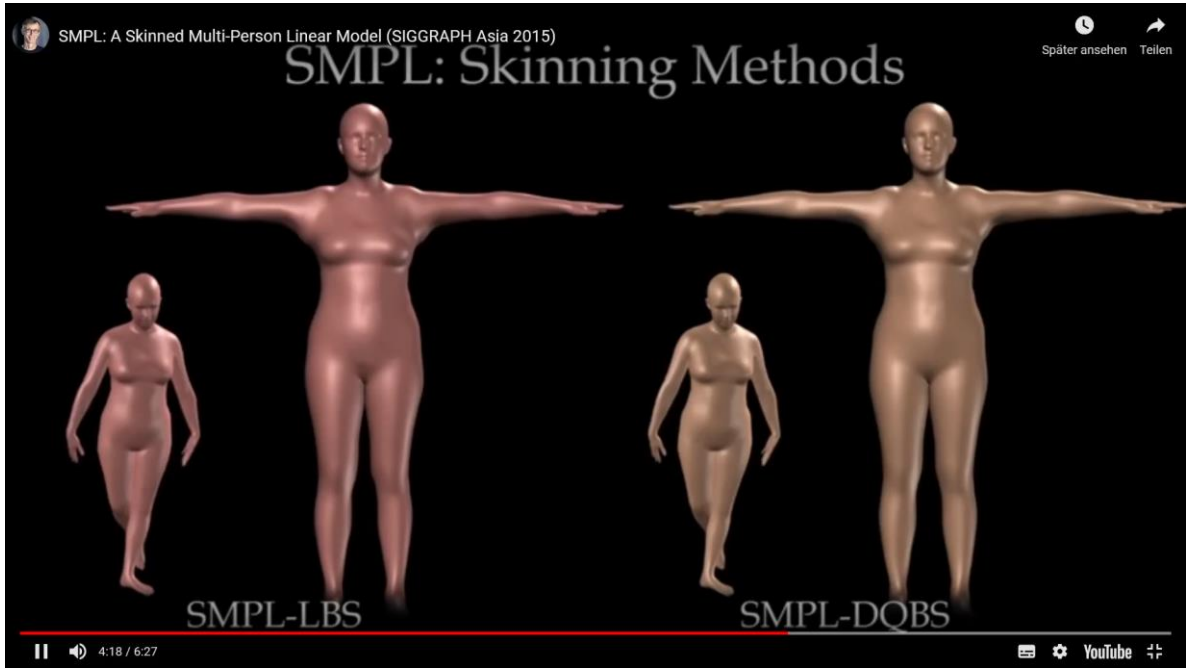
Geometric Approaches

- Main idea
  - Mesh + skeleton + weights
  - define Skinning Transformation that deforms skin based on geometric heuristics
  - Linear blend skinning
  - Dual quaternion skinning
- Advantages
  - Simple & efficient
- Disadvantages
  - Not physically correct
  - Yields some artifacts

www.youtube.com/watch?v=LUOJccOZfWQ

A Skinned Multi-Person Linear Model

- SMPL-Approach combines scanning and skinning transformation
- Inverse skinning transformation is used to disentangle pose from shape and muscle deformations
- Linear models are used in reference pose to model shape variation and muscle deformations



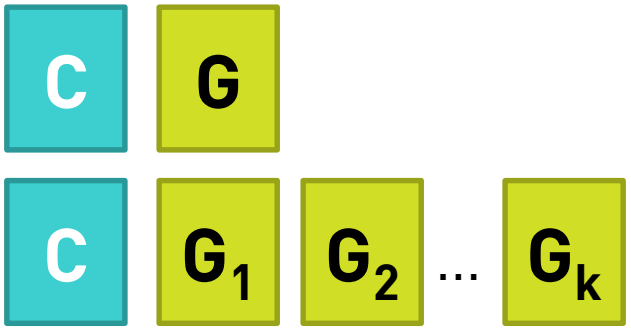
Loper, Matthew, et al. "SMPL: A skinned multi-person linear model." *ACM transactions on graphics (TOG)* 34.6 (2015): 1-16.

https://smpl.is.tue.mpg.de/

SKINNED MESHES

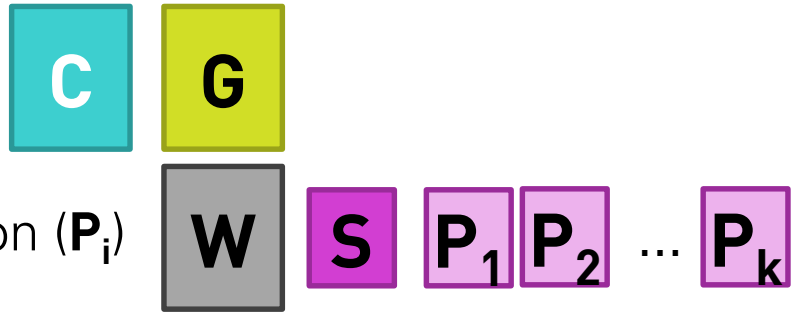
Comparison Dynamic Geometry

- Keyframe based shape animation
  - design surface model (geometry plus connectivity)
  - deform model per key-frame (static connectivity plus per keyframe geometry)



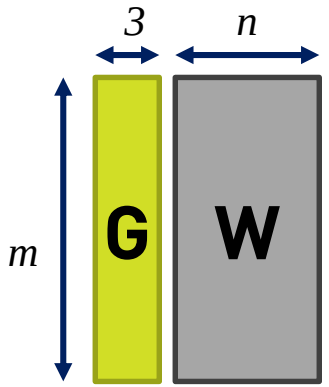
- physically based animation
  - design model / scene with physical parameters
  - compute keyframe based shape animation from laws of physics

- character animation
  - design surface model (**C**+**G**)
  - design skeleton **S** with bones
  - attach vertices to bones (**W**)
  - animate keyframe poses of skeleton ($\mathbf{P}_i$)
  - compute deformed surface model



Storing a Skinned Mesh

- mesh vertices in rest pose:
  $j = 1 \ldots m \qquad \boldsymbol{p}_j^0$
- skeleton joints
  $i = 1 \ldots n$
- per joint vertex weights
  $n \cdot m : w_{ij} \in [0,1]$
- per time step τ store pose as joint parameters

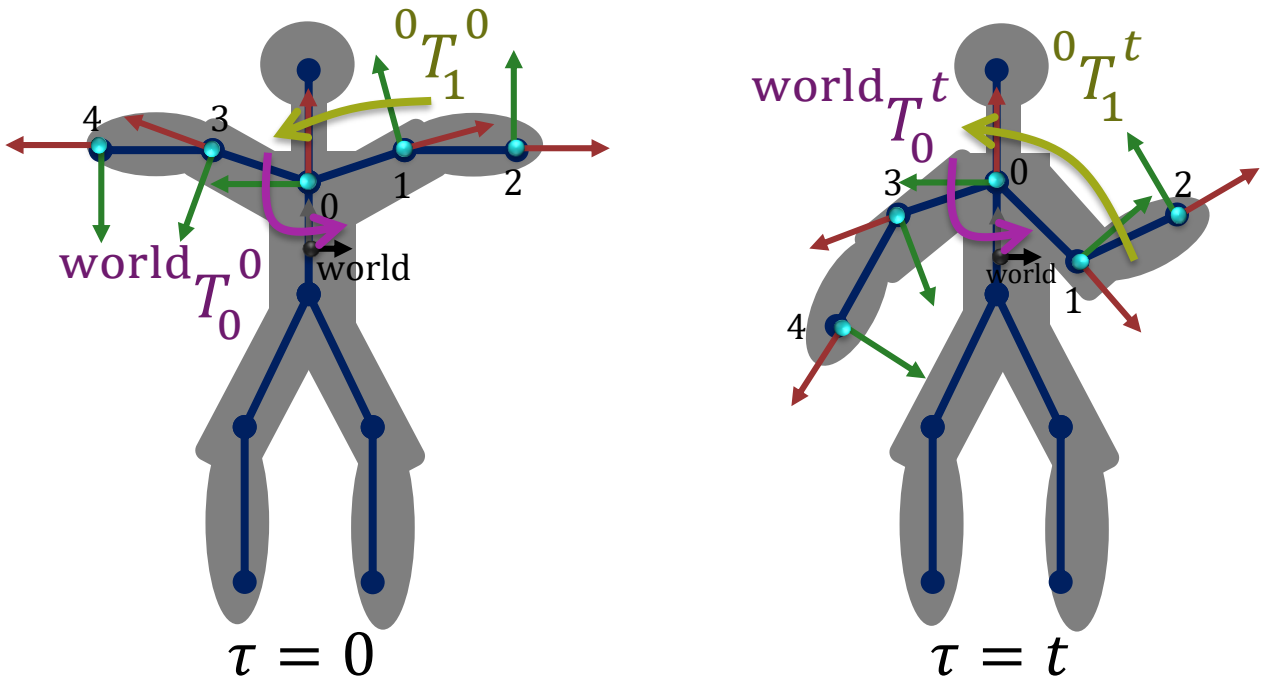


typically W should be sparse
for example no more than
4 weights per row unequal 0

Mesh Skinning Overview

- **preparation (rigging):** align skeleton with skin mesh in reference pose ($\tau \equiv 0$), and define vertex weights

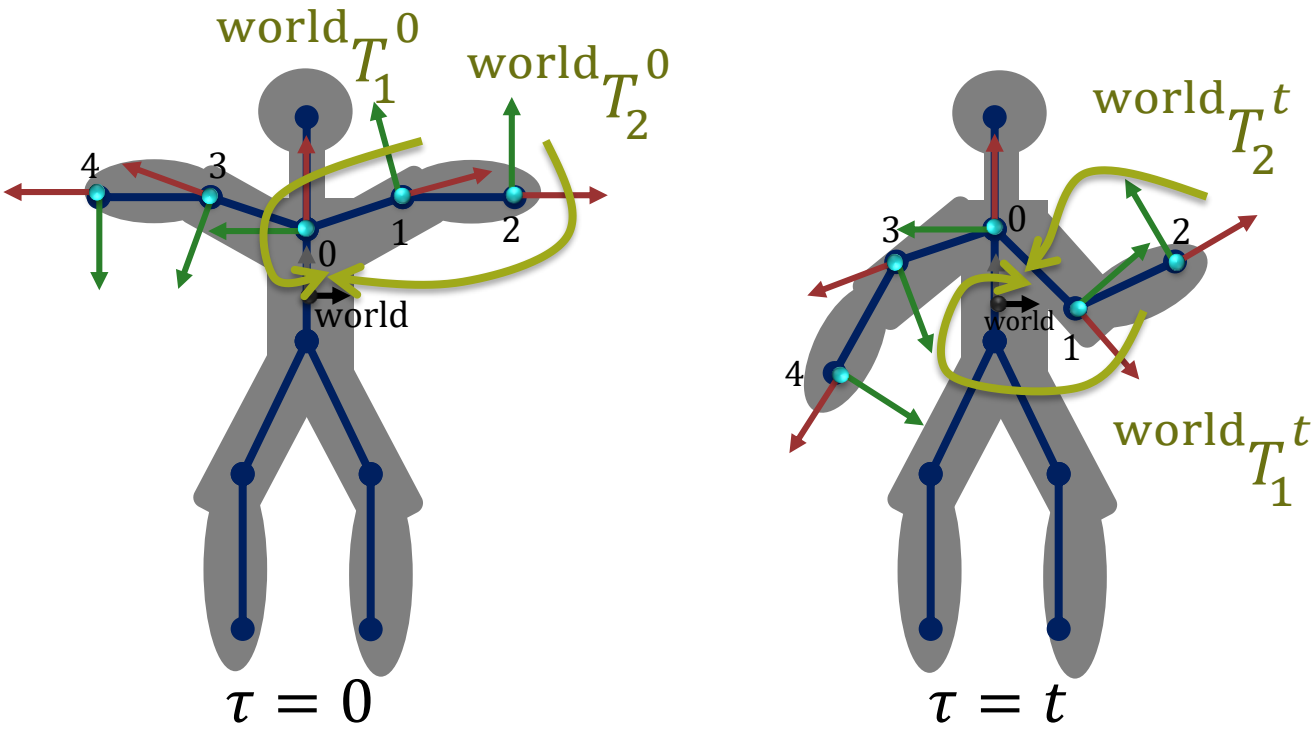
1. **define pose:** per frame $\tau = t$ design skeleton pose with local joint transformations ${}^{p(i)}T_i^{\tau}$ & root transform ${}^{\text{world}}T_0^{\tau}$



Mesh Skinning Overview

2. compute joint to world transformations:

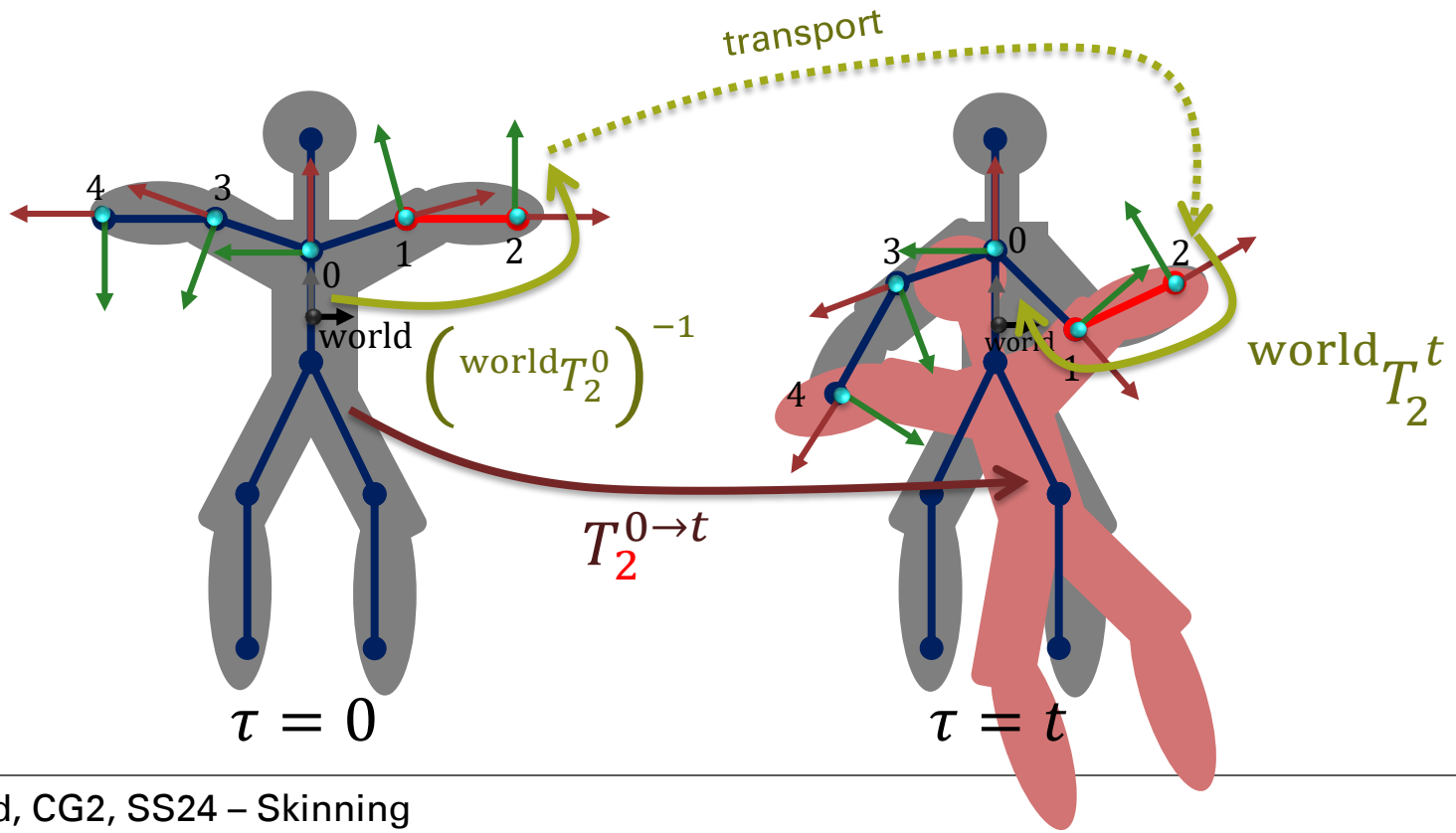
$$ {}^{\text{world}}T_i^{\tau} = {}^{\text{world}}T_{p(i)}^{\tau} \; {}^{p(i)}T_i^{\tau} $$



Mesh Skinning Overview

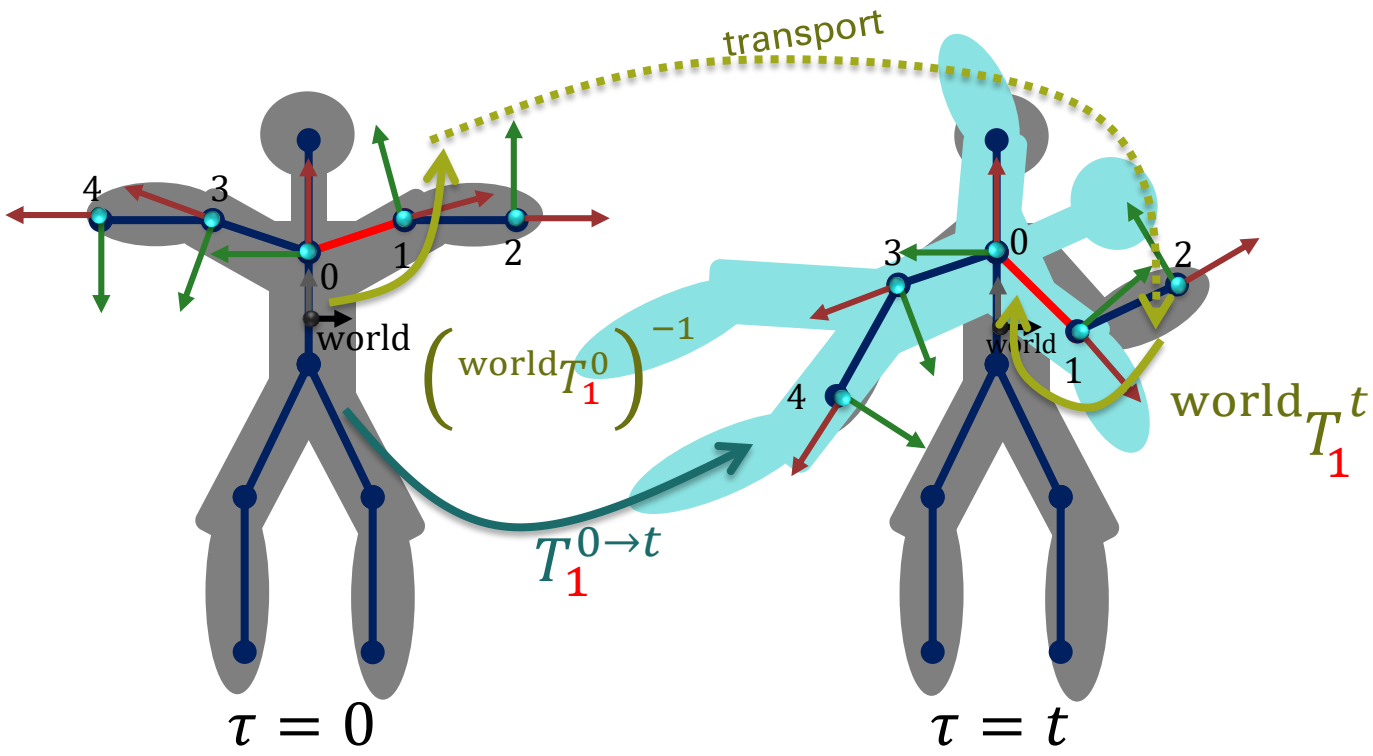
3. **compute joint pose transformations** that transform from reference pose world coordinates to current pose world coordinates:

$$T_i^{0\to t} = {}^{\text{world}}T_i^t \left({}^{\text{world}}T_i^0\right)^{-1} = {}^{\text{world}}T_i^t \cdot {}^{i}T_{\text{world}}^0$$



Mesh Skinning Overview

- A vertex is influenced by all joint pose transformations where the vertex weight is larger than zero.



Linear Blend Skinning Transform

- The Linear Blend Skinning Transformation maps skin mesh from rest to current pose:

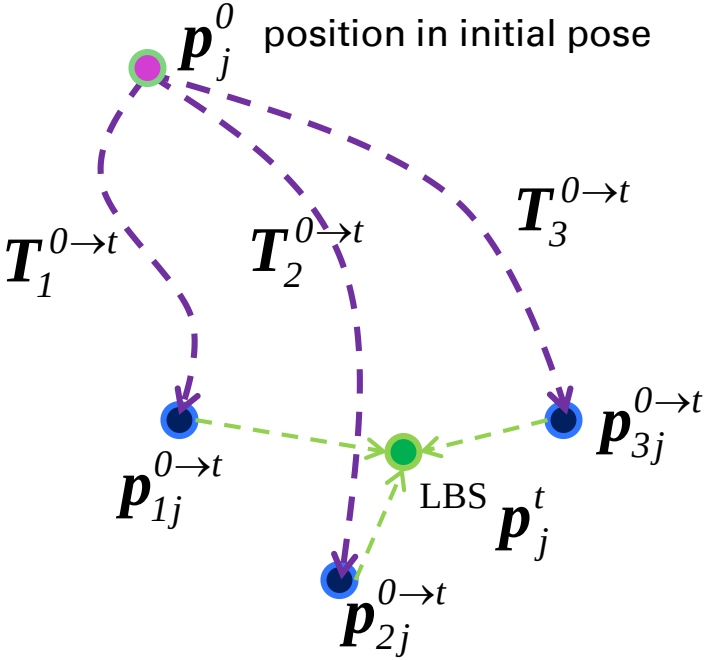
$$\boldsymbol{P}_t = \boldsymbol{M}_{LBS}(q_{ik}, \boldsymbol{W}, \boldsymbol{P}_0)$$

by weighted average over joint pose transformation results:

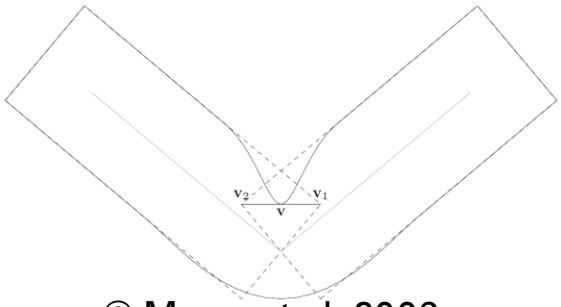
$$\forall j: {}^{\text{LBS}}\tilde{\boldsymbol{p}}_j^t = \sum_{i=1}^{n} w_{ij}\tilde{\boldsymbol{p}}_{ij}^{0\to t}$$

and $\sum_{i=1}^{n} w_{ij} = 1$

- This technique is also called skeletal subspace deformation (SSD)
- but linear vertex blending leads to elbow collapse and Candy wrapper artefacts due to loss of volume.



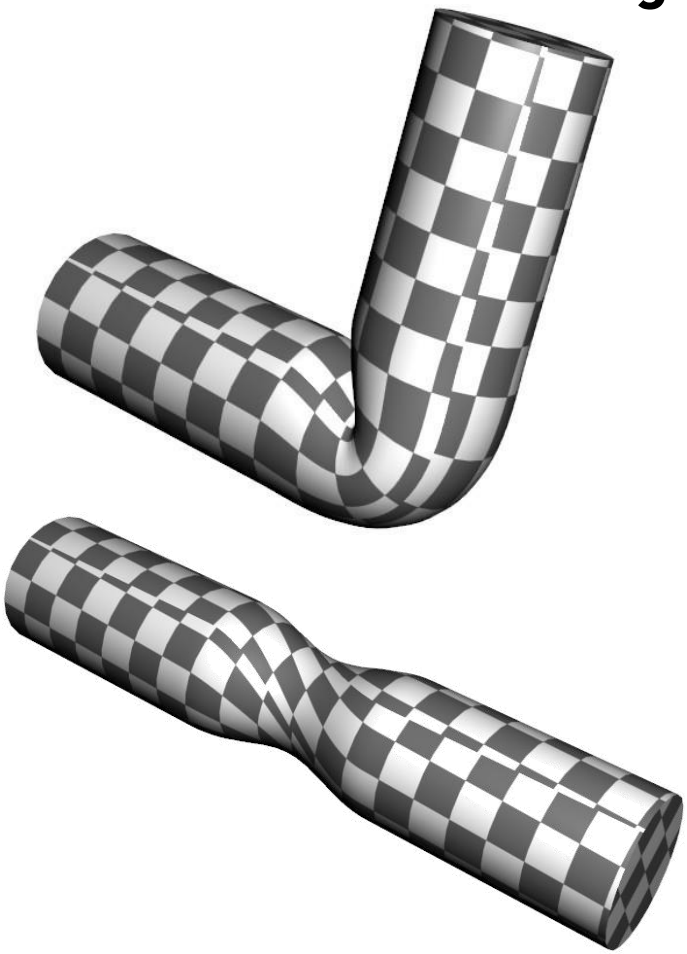
$$\tilde{\boldsymbol{p}}_{ij}^{0\to t} := \boldsymbol{T}_i^{0\to t}\tilde{\boldsymbol{p}}_j^0$$



© Merry et al. 2006

LBS artefacts

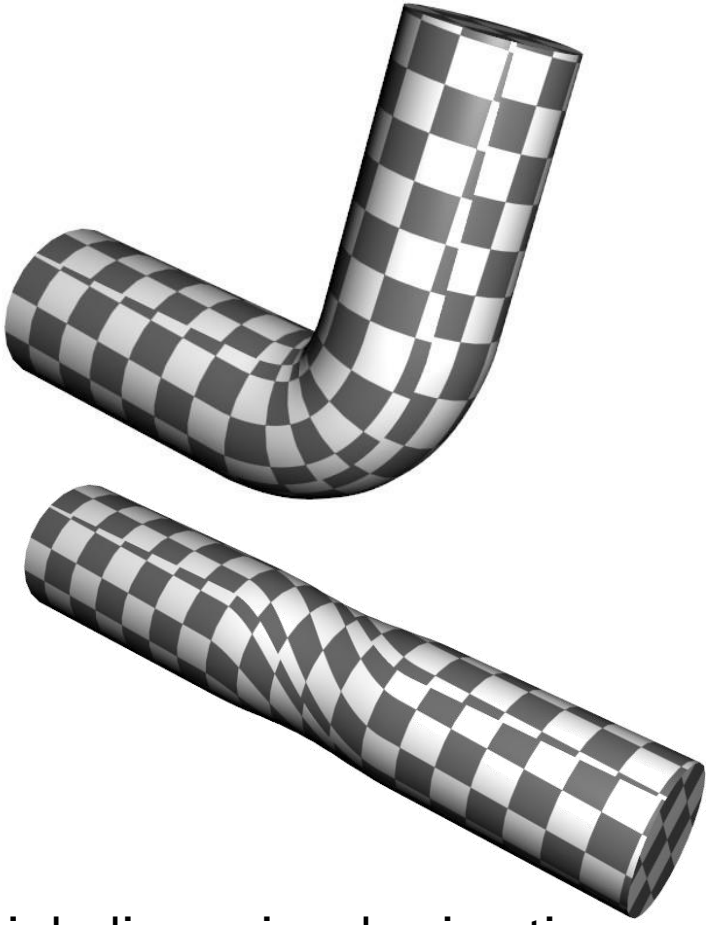
linear blend skinning



elbow collapse artefact

candy wrapper artefact

animation space blending



high dimensional animation space to be learned from examples

all figures © Merry et al. 2006

INTERPOLATION OF TRANSFORMATIONS

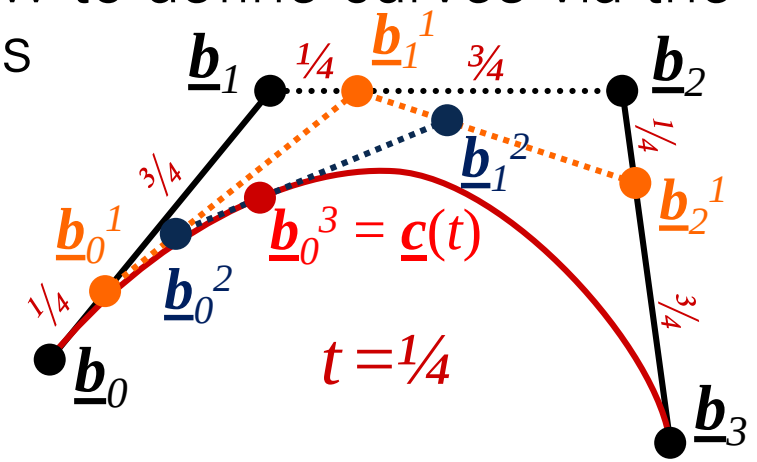
Interpolation by affine combination

- An affine combination can interpolate between two points

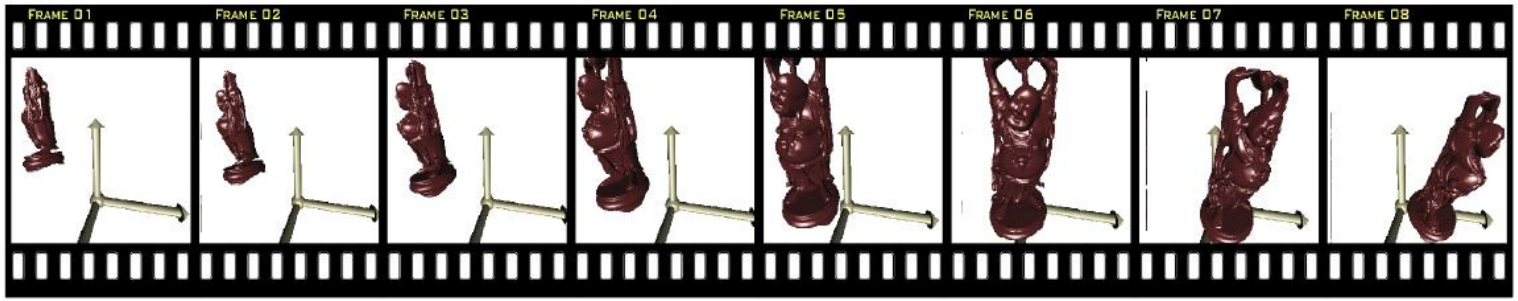
$$\underline{c}(t) = (1-t)\underline{p} + t\underline{q} = \underline{p} + t(\underline{q} - \underline{p})$$

- Iterated affine combinations allow to define curves via the deCasteljau or deBoor algorithms

$$\underline{c}(t) = \underline{b}_0^g \qquad \underline{b}_i^0 = \underline{b}_i$$
$$\underline{b}_i^r = (1-t)\underline{b}_i^{r-1} + t\underline{b}_{i+1}^{r-1}$$



- Generalization of affine combinations allow to build curves over transformations, which can be used for camera animation:



© Alexa 2002

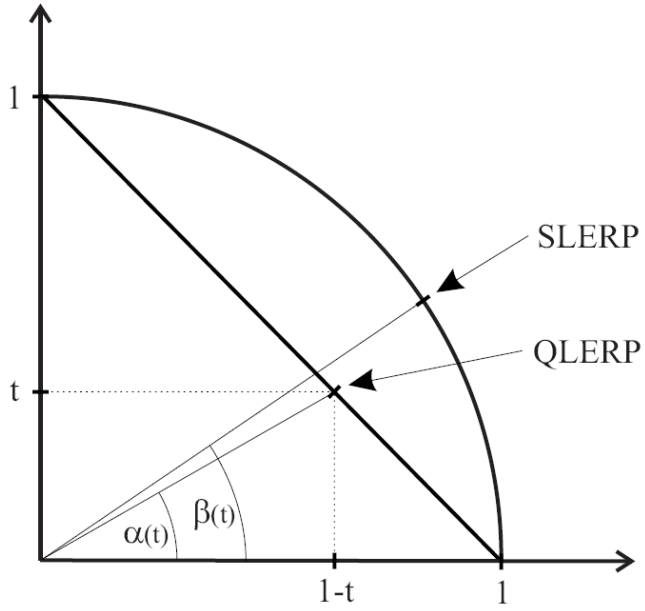
Interpolation of Quaternions

- One can interpolate two unit quaternions q_1 and q_2 with the **SLERP** operation, resulting in an interpolation of the corresponding rotations with respect to angle

$$q_2' = \begin{cases} +q_2 & \langle q_1, q_2 \rangle \geq 0 \\ -q_2 & \text{otherwise} \end{cases}$$

$$^{\text{SLERP}}q(t) = \frac{\sin((1-t)\theta)q_1 + \sin(t\theta)q_2'}{\sin\theta}$$

with $\cos\theta = \langle q_1, q_2' \rangle$



- In linear interpolation of quaternions the rotation speed varies slightly

$$^{\text{QLERP}}q(t) = \frac{(1-t)q_1 + tq_2}{\|(1-t)q_1 + tq_2\|}$$

Interpolation of transformations

- Given 2 transformations as homogeneous matrices $\widetilde{\boldsymbol{M}}_0$ and $\widetilde{\boldsymbol{M}}_1$ we look for in between matrices $\widetilde{\boldsymbol{M}}_t$ for $t \in [0,1]$
- define relative transformation $\widetilde{\boldsymbol{M}}_{01} = \widetilde{\boldsymbol{M}}_0^{-1}\widetilde{\boldsymbol{M}}_1$

Interpolation Approach 1:

- decompose matrix into $\widetilde{\boldsymbol{M}}_{01} = \widetilde{\boldsymbol{P}}\widetilde{\boldsymbol{T}}\widetilde{\boldsymbol{R}}\left[\widetilde{\boldsymbol{N}}\right]\widetilde{\boldsymbol{O}}\widetilde{\boldsymbol{S}}\widetilde{\boldsymbol{O}}^T$

perspective $\widetilde{\boldsymbol{P}}(\vec{\boldsymbol{p}}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p_x & p_y & p_z & 1 \end{pmatrix}$ translate $\widetilde{\boldsymbol{T}}(\vec{\boldsymbol{t}}) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$ rotate $\widetilde{\boldsymbol{R}}(\hat{\boldsymbol{n}}, \alpha)$ reflect $\widetilde{\boldsymbol{N}}$ orient $\widetilde{\boldsymbol{O}}$ scale $\widetilde{\boldsymbol{S}}(\vec{\boldsymbol{s}}) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

- interpolation works only if reflection $\widetilde{\boldsymbol{N}}$ not present:

$$\widetilde{\boldsymbol{M}}_t = \widetilde{\boldsymbol{M}}_0\widetilde{\boldsymbol{P}}(t \cdot \vec{\boldsymbol{p}})\widetilde{\boldsymbol{T}}\left(t \cdot \vec{\boldsymbol{t}}\right)\widetilde{\boldsymbol{R}}(\hat{\boldsymbol{n}}, t \cdot \alpha)\widetilde{\boldsymbol{O}}\widetilde{\boldsymbol{S}}(\vec{\boldsymbol{s}}^t)\widetilde{\boldsymbol{O}}^T$$

Interpolation Approach 2:

- use matrix powers: $\widetilde{\boldsymbol{M}}_t = \widetilde{\boldsymbol{M}}_0\left(\widetilde{\boldsymbol{M}}_{01}\right)^t$
  (see matrix blending for existence of matrix powers)

LOG MATRIX BLENDING

Blending of Special Cases

Translations

- in case of pure translations, the blending is very simple
- let $\vec{\boldsymbol{t}}_i$ be the translation vectors, which can be blended perfectly according to

$$\bar{\vec{\boldsymbol{t}}} = \sum_i^n w_i \vec{\boldsymbol{t}}_i$$

Scaling

- in case of pure scalings, the blending is also simple
- let $s_{\alpha \in \{x,y,z\},i}$ be the scaling factors on the diagonal components of the scaling matrices, then perfect blending is achieved by

$$\bar{s}_\alpha = \prod_{i=1}^n s_{\alpha,i}^{w_i}$$

Rotation in 2D

- in case of 2D rotations with angles α_i perfect

blending is achieved by

$$\bar{\alpha} = \sum_i^n w_i \alpha_i$$

Log Blending

- **Basic idea:** the log function can be used to transform the product operation into a sum:

$$a \cdot b = \exp(\log a + \log b)$$

- We call all numbers α of the form $\log a$ the log space.
- This works only for positive numbers a. Log space on the other hand contains also negative numbers.
- Number 1 corresponds to 0 in log space, which is the neutral element and is special.
- We can choose any positive number r to map to the neutral element by defining

$$\log_r a := \log(a/r) = \log a - \log r$$
$$\exp_r \alpha := \exp(\alpha + \log r) = r \cdot \exp \alpha$$

- Careful! Here r is not a different basis of the logarithm.
- Weighted log blending of n numbers a_i with weights w_i around the reference r is finally defined as

$$\bar{a} = \exp_r(\sum_i^n w_i \log_r a_i)$$

Log Matrix Blending (Alexa 2002)

- In order to apply the log blending idea to matrices one has to clarify when logarithm and exponential of a matrix exist:
- for real matrices logarithms exist if and only if roots exist
- roots of real matrix exist if
  - matrix is invertible,
  - determinant is positive and
  - all real negative eigenvalues have even multiplicity

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}^{\frac{1}{2}} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

real square root exists

$$\begin{pmatrix} -2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}^{\frac{1}{2}} \in \mathbb{C}^{3\times 3} \setminus \mathbb{R}^{3\times 3}$$

real square root does not exist

- If logarithms exist of $\boldsymbol{A}$ and $\boldsymbol{B}$ one can proof the identity

$$\exp(\log \boldsymbol{A} + \log \boldsymbol{B}) = \lim_{n\to\infty} \left(\boldsymbol{A}^{\frac{1}{n}} \boldsymbol{B}^{\frac{1}{n}}\right)^n$$

- This means that $\boldsymbol{A}$ and $\boldsymbol{B}$ are split into infinitesimally small parts which are interwoven to a commutative result.

Log Matrix Blending (Alexa 2002)

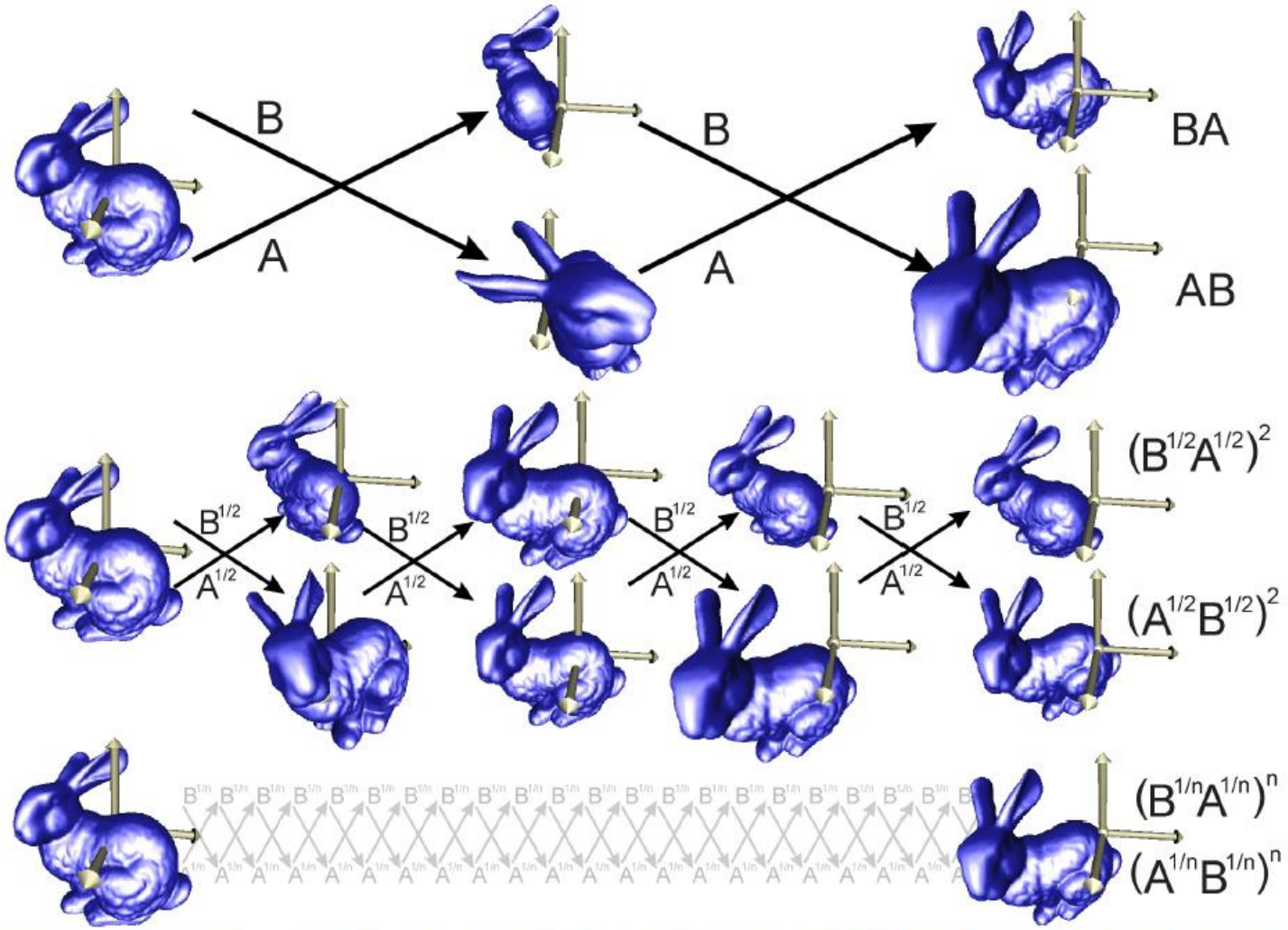


Figure 3: The addition of transformations. Given two transformations, A and B, applying one after the other (i.e. multiplying the matrices) generally leads to different results depeneding on the order of the operations. By performing n-th parts of the two transformations in turns the difference of the two orders becomes smaller. The limit $n \to \infty$ could be understood as performing both transformations concurrently. This is the intuitive geometric definition of a commutative addition for transformations based on the matrices.

Log Matrix Blending (Alexa 2002)

- Given some reference matrix $\boldsymbol{M_0}$ one can extend the exp and log functions to

$$\log_{\boldsymbol{M_0}} \boldsymbol{M} = \log \boldsymbol{M} - \log \boldsymbol{M_0}$$
$$\exp_{\boldsymbol{M_0}} \boldsymbol{m} = \exp(\boldsymbol{m} + \log \boldsymbol{M_0}) = \boldsymbol{M_0} \cdot \exp \boldsymbol{m}$$

- and define log matrix blending as

$$\exp_{\boldsymbol{M_0}}\left(\sum_{i=1}^{n} w_i \log_{\boldsymbol{M_0}} \boldsymbol{M_i}\right)$$

Properties when applied to rigid body transformations

- result is always a valid rigid body transformation
- interpolates between two rotations with uniform angular speed
- interpolation between two rotations is not necessarily along the shortest path and does not correspond to rotation around a static axis.

Log Matrix Blending Discussion

- for implementation of log and exp functions for matrix class see Alexa's paper (iterative methods with standard functions only)
- More discussion on log matrix blending in Bloom et al. *Errors and Omissions in Marc Alexa's "Linear Combination of Transformations"*
- They point out, that rotations are represented by an infinite number of matrices in log space that correspond to rotation by angles $\alpha, \alpha + 2\pi, \alpha + 4\pi, \ldots$. This complicates spline interpolation and analysis techniques like SVD significantly.

SPHERICAL BLENDING

Spherical Averages (Buss 2001)

- given a unit quaternion q, one can apply an exponential map that preserves geodesic distances on the unit sphere to q
- repeated conversion to log space, averaging in log space and conversion back to quaternion space converges to the optimal quaternion average with the uniform speed property
- (has not been used in skinning)

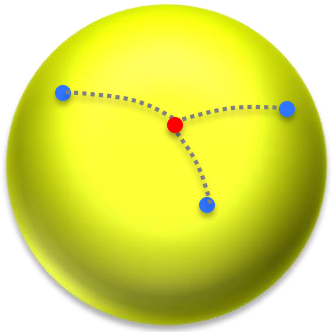
$$q^{k=0} = \sum_i w_i q_i$$

repeat till convergence

$$u_i^k = \log q_i - \log q^k$$

$$u^k = \sum_i w_i u_i^k$$

$$q^{k+1} = q^k \cdot \exp u^k$$

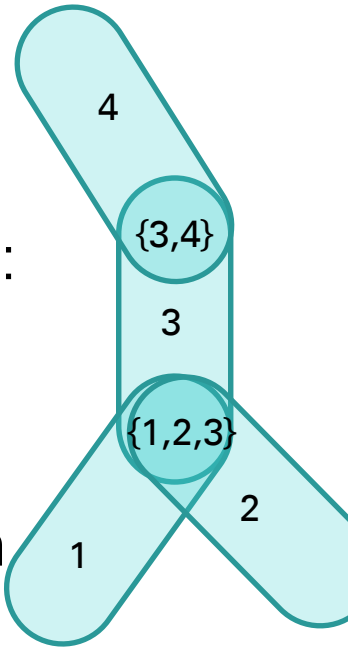


Spherical Blend Skinning (Kavan 2005)

- Split joint pose transformations into a quaternion and a translation vector $\widetilde{\boldsymbol{T}}_i^{0\to t} \to \pm q_i, \vec{\boldsymbol{t}}_i$
- Choose sign of quaternions to make scalar products positive: $\forall i: q_i' = \mathrm{sgn}(\langle q_i, q_1 \rangle) \cdot q_i$
- Per vertex j find index tuple $I_{\alpha_j} = \{i_1, i_2, \ldots, i_k\}$ where $w_{ij} > 0$ and 0 otherwise and blend translation vectors and quaternions independently:

$$\bar{q}_j = \mathrm{normalize}\left(\sum_{i \in I_{\alpha_j}} w_{ij} q_i'\right), \vec{\boldsymbol{t}}_j = \sum_{i \in I_{\alpha_j}} w_{ij} \vec{\boldsymbol{t}}_i$$

- For each index tuple $I_\alpha = \{i_1, i_2, \ldots, i_k\}$ find rotation center $\underline{\boldsymbol{r}}_\alpha^0$ mapped to $\underline{\boldsymbol{r}}_{\alpha,i}^t = \widetilde{\boldsymbol{T}}_{i \in I_\alpha}^{0\to t} \underline{\boldsymbol{r}}_\alpha^0$ such that the $\underline{\boldsymbol{r}}_{\alpha,i}^t$ are as close as possible.
- Finally: ${}^{\mathrm{SBS}}\underline{\boldsymbol{p}}_j^t := \bar{q}_j \left(\underline{\boldsymbol{p}}_j^0 - \underline{\boldsymbol{r}}_{\alpha_j}^0\right) \bar{q}_j^* + \sum_{i \in I_\alpha} w_{ij} \underline{\boldsymbol{r}}_{\alpha,i}^t + \vec{\boldsymbol{t}}_j$



at revolute joints $\underline{\boldsymbol{r}}_\alpha^0$ are on joint axes

Spherical Vertex Blending – Comparison

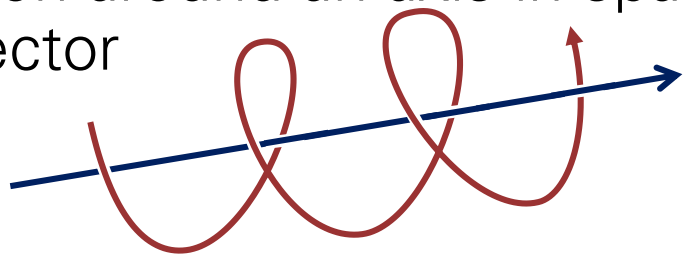
- Spherical Blend Skinning vs. Linear Blend Skinning

www.youtube.com/watch?v=ksDgnOYzWM0

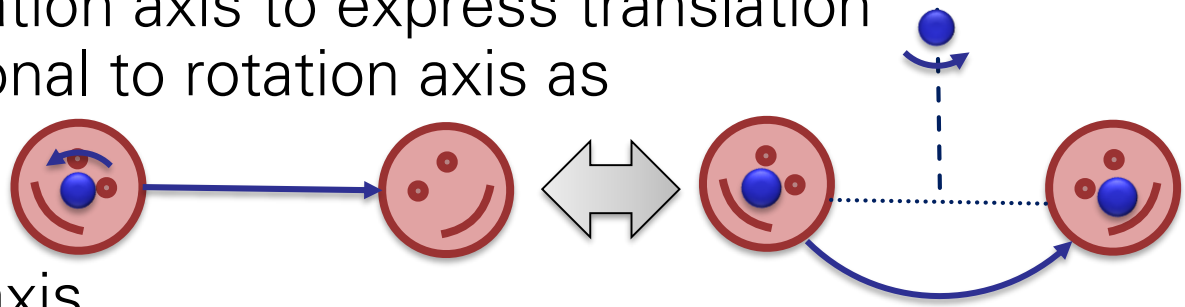
SCREW MOTIONS & DUAL QUATERNION BLENDING

Rigid Body Transformations

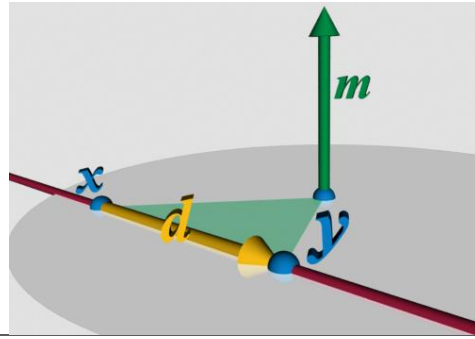
- **Chasle's/Mozzi's/Cauchy's screw theorem:** Any rigid body transformation $\boldsymbol{R}\underline{\boldsymbol{p}} + \vec{\boldsymbol{t}}$ can be represented as a screw motion, i.e. a rotation around an axis in space that is parallel to translation vector



- Proof: translate rotation axis to express translation component orthogonal to rotation axis as part of the rotation



- Representation of axis
  - two positional vectors $\vec{\boldsymbol{x}}$ and $\vec{\boldsymbol{y}}$, or
  - Plücker coordinates: $\{\vec{\boldsymbol{d}}:\vec{\boldsymbol{m}}\}$ with 2 ortho. vectors: direction $\vec{\boldsymbol{d}} = \vec{\boldsymbol{y}} - \vec{\boldsymbol{x}}$, momentum $\vec{\boldsymbol{m}} = \vec{\boldsymbol{x}} \times \vec{\boldsymbol{y}}$
- Plus rotation angle θ and translation offset d



Dual Coordinates

- Dual numbers have a real and a dual component and are defined similar to complex numbers (denoted with hat):

$$\hat{a} := a + \varepsilon a_\varepsilon$$

- With dual part a_ε and dual base ε similar to complex i but with

$$\varepsilon^2 = 0 !!$$

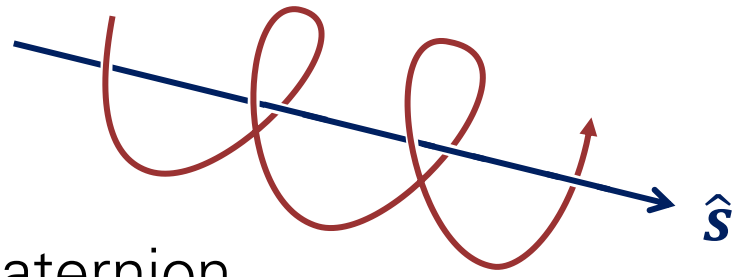
- multiplication $(a + \varepsilon a_\varepsilon)(b + \varepsilon b_\varepsilon) = ab + \varepsilon(ab_\varepsilon + a_\varepsilon b)$
- conjugation $\bar{\hat{a}} = a - \varepsilon a_\varepsilon$
- inverse $a \neq 0 \Rightarrow \hat{a}^{-1} = \frac{1}{a} - \varepsilon \frac{a_\varepsilon}{a^2}$
- square root $\sqrt{a + \varepsilon a_\varepsilon} = \sqrt{a} + \varepsilon \frac{a_\varepsilon}{2\sqrt{a}}$

Dual Quaternions

- dual quaternion $\hat{q} = \hat{w} + i\hat{x} + j\hat{y} + k\hat{z}$ with $i\varepsilon = \varepsilon i, \ldots$
  $= w + \varepsilon w_\varepsilon + ix + i\varepsilon x_\varepsilon + jy + j\varepsilon y_\varepsilon + kz + k\varepsilon z_\varepsilon$
  $= q + \varepsilon q_\varepsilon$
- conjugations $\hat{q}^* = \hat{w} - i\hat{x} - j\hat{y} - k\hat{z}$ $\quad \bar{\hat{q}} = q - \varepsilon q_\varepsilon$
- norm $\|\hat{q}\| = \sqrt{\hat{q}\hat{q}^*} = \|q\| + \varepsilon \frac{\langle q, q_\varepsilon \rangle}{\|q\|}$ with $\|\hat{q}_1 \hat{q}_2\| = \|\hat{q}_1\| \cdot \|\hat{q}_2\|$
- inverse quaternion $\hat{q}^{-1} = \hat{q}^* / \|\hat{q}\|^2$
- for unit dual quaternions dual and non dual part are orthogonal $\|\hat{q}\| = 1 \Rightarrow \|q\| = 1 \quad \wedge \quad \langle q, q_\varepsilon \rangle = 0$
- dual quaternion of 3d vector $\underline{v} \cong 1 + \varepsilon(v_x i + v_y j + v_z k)$
- transformation of vector with dual quaternion $\underline{v}' = \hat{q}\underline{v}\bar{\hat{q}}^*$

Unit Dual Quaternions

- when applying functions to dual numbers we get:
  - real part: function applied to real part
  - dual part: derivative applied to real part times dual part
- for trigonometric functions we get
  - $\sin\hat{\theta} = \sin\theta + (\cos\theta)\theta_\varepsilon\varepsilon$
  - $\cos\hat{\theta} = \cos\theta - (\sin\theta)\theta_\varepsilon\varepsilon$
  - $\tan\hat{\theta} = \tan\theta - \frac{\theta_\varepsilon}{\cos^2\theta}\varepsilon$



- screw motion form of unit dual quaternion with dual angle $\hat{\theta}$ and dual axis vector $\hat{\boldsymbol{s}}$:

$$\hat{q} = \left(\cos\frac{\hat{\theta}}{2}, \sin\frac{\hat{\theta}}{2}\hat{\boldsymbol{s}}\right)$$

with the individual components:

- $\frac{\theta}{2}$ …half rotation angle; $\frac{\theta_\varepsilon}{2}$ … half translation offset
- $\boldsymbol{s}$ … axis direction vector; $\boldsymbol{s}_\varepsilon$ … axis moment vector

Dual Quaternion Blending(Kavan 2007)

- Convert joint pose transformation into dual quaternions $\widetilde{\boldsymbol{T}}_i^{0\to t} \to \pm\hat{q}_i$
- Choose sign of dual quaternions to make scalar products positive:$\forall i: \hat{q}_i' = \mathrm{sgn}(\langle \hat{q}_i, \hat{q}_1 \rangle) \cdot \hat{q}_i$
- Per vertex j find index tuple $I_{\alpha_j} = \{i_1, i_2, \ldots, i_k\}$ where $w_{ij} > 0$ and 0 otherwise and blend dual quaternions:

$$\hat{q}_j' = \mathrm{normalize}\left(\sum_{i \in I_{\alpha_j}} w_{ij}\hat{q}_i'\right)$$

- Finally:

$$^{\mathrm{DQS}}\underline{\boldsymbol{p}}_j^t := \hat{q}_j' \underline{\boldsymbol{p}}_j^0 \bar{\hat{q}}_j'^*$$

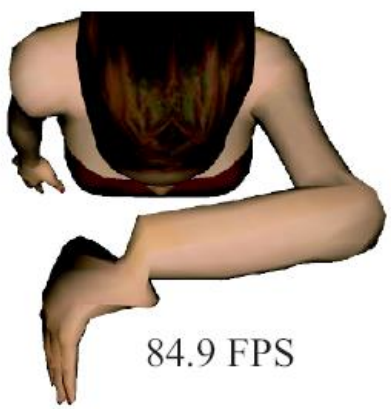
Dual quaternion Linear Blending

Properties

- always computes a valid rigid body transformation
- invariant under coordinate transformations (and choice of rotation center)
- interpolates two rigid transformations along the shortest path
- very fast computation

Missing Property

- does not interpolate transformations with uniform speed



Log-matrix Blending

Dual Quaternions

55.1 FPS

Spherical Blend Skinning

122 FPS

Dual Quaternions

references

- Jeremiah van Oosten, Understanding Quaternions, http://3dgep.com/?p=1815
- John Hart, Quaternion Demonstrator, https://graphics.stanford.edu/courses/cs348c-95-fall/software/quatdemo
- Convert from rotation matrix to quaternion, http://www.cg.info.hiroshima-cu.ac.jp/~miyazaki/knowledge/teche52.html
- Ben Kenwright: A Beginners Guide to Dual-Quaternions, http://wscg.zcu.cz/wscg2012/short/a29-full.pdf

TRANSFORMATION OF NORMALS

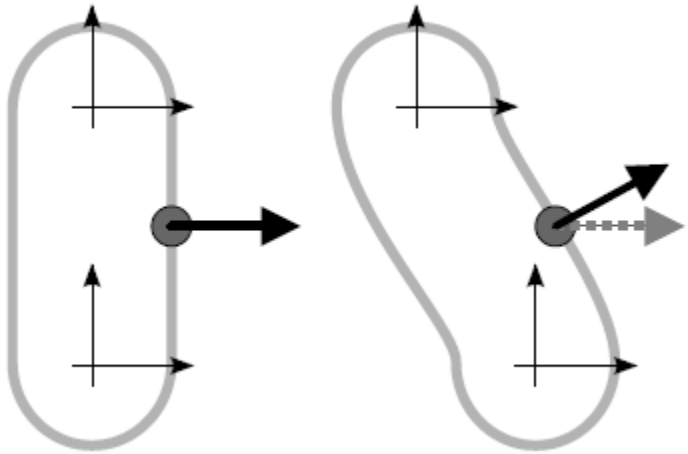
Transformation of Normals

- In 2014 Tarini et al. realized that the blended transformations of all blending schemes can't be used to transform vertex normals correctly.

$$^{\mathrm{LBS}}\widetilde{\boldsymbol{T}}_j^{0\to t} = \sum_{i=1}^{n} w_{ij}\widetilde{\boldsymbol{T}}_i^{0\to t}, \; ^{\mathrm{LBS}}\widetilde{\boldsymbol{p}}_j^{t} = {}^{\mathrm{LBS}}\widetilde{\boldsymbol{T}}_j^{0\to t}\widetilde{\boldsymbol{p}}_j^{0}$$

$$^{\mathrm{DQS}}\hat{q}_j^{0\to t} = \sum_{i=1}^{n} w_{ij}\hat{q}'^{0\to t}_i \Big/ \left\| \sum_{i=1}^{n} w_{ij}\hat{q}'^{0\to t}_i \right\|$$

- In the example on the right, there is only a translation between the joint frames such that all blending methods stick to translations only. The normal is bent due to the change of vertex weights and not through a rotation.



- They propose solution that is a bit too complicated to discuss in detail here (see paper)

Transformation of Normals

Accurate and Efficient Lighting for Skinned Models

Marco Tarini[1,2]
Daniele Panozzo[3]
Olga Sorkine-Hornung[3]

[1] Università dell'Insubria, Varese
[2] ISTI-CNR Pisa
[3] ETH Zurich

https://vimeo.com/85267396

REFERENCES

References

- Marc Alexa. 2002. Linear combination of transformations. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '02). ACM, New York, NY, USA, 380-387. DOI=10.1145/566570.566592
- Ladislav Kavan and Jiří Žára. 2005. Spherical blend skinning: a real-time deformation of articulated models. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games* (I3D '05). ACM, New York, NY, USA, 9-16. DOI=10.1145/1053427.1053429
- Bruce Merry, Patrick Marais , James Gain. 2006. Animation space: a truly linear framework for character animation. *ACM Transactions on Graphics* 25(4):1400-1423. http://pubs.cs.uct.ac.za/archive/00000373/
- Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2007. Skinning with dual quaternions. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games* (I3D '07). ACM, New York, NY, USA, 39-46. DOI=10.1145/1230100.1230107
- Marco Tarini, Daniele Panozzo, Olga Sorkine-Hornung, Accurate and Efficient Lighting for Skinned Models, Eurographics 2014