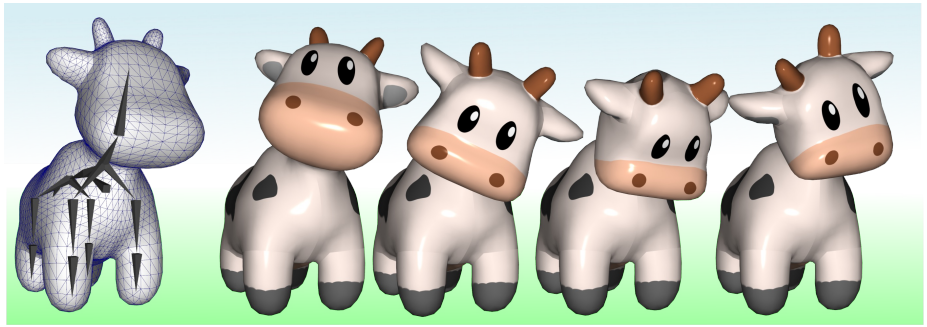
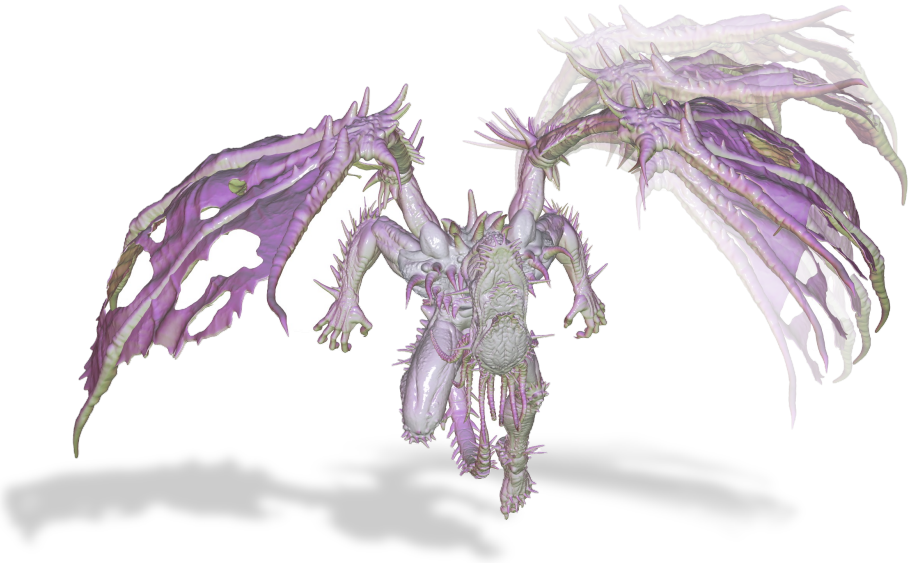
Real-time Expressive Dynamic Animation of 3D Virtual Character



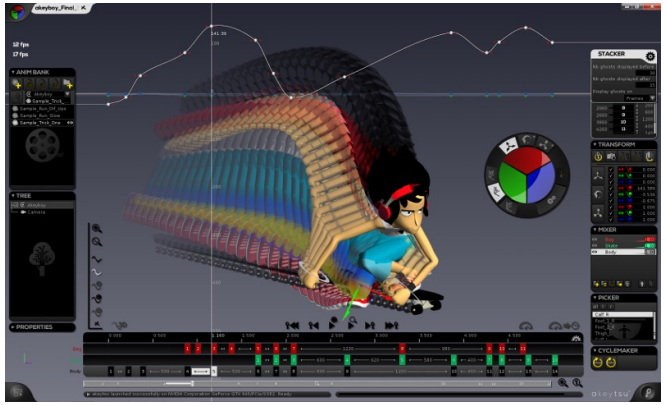


Damien Rohmer

Damien.rohmer@polytechnique.edu

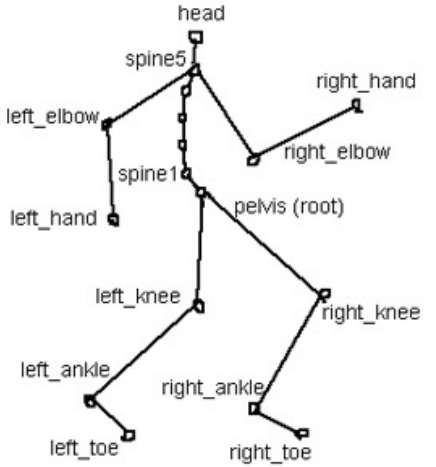
Terminology

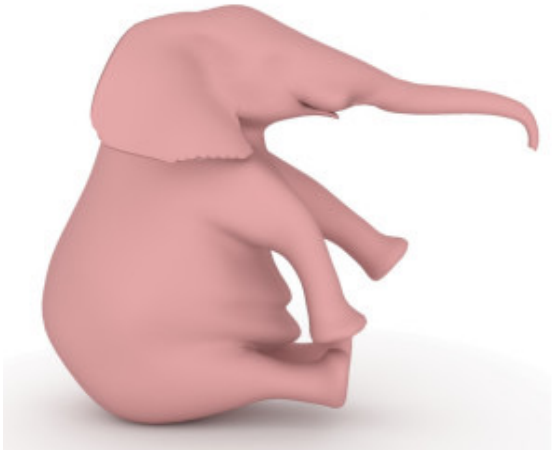
Real-time Expressive Dynamic **Animation of 3D Virtual Character**



Character body

- its pose *(often a skeleton)*
- its shape/skin deformation





(Not include Facial Animation -> its own topic/research study)

Terminology

Real-time **Expressive Dynamic Animation** of 3D Virtual Character

In "facial expression": convey an emotion/feeling

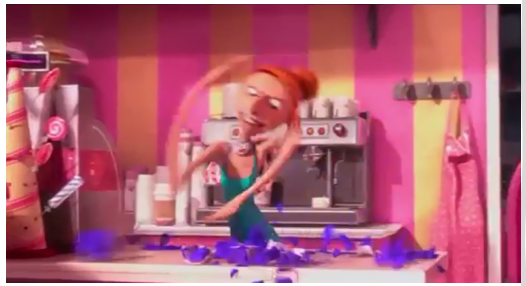
Extended to **character animation**

- **energy** (dynamics, velocity, weight, etc),
- **material property** (elasticity vs rigidity)

Often exaggerated deformations – Cartoon-like

Visible even on static images





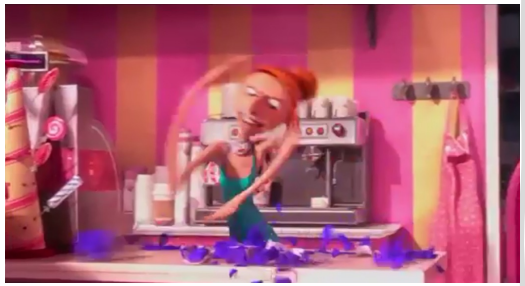


Terminology

Real-time **Expressive Dynamic Animation** of 3D Virtual Character

In "facial expression": convey an emotion/feeling





Extended to **character animation**

- **energy** (dynamics, velocity, weight, etc),
- **material property** (elasticity vs rigidity)



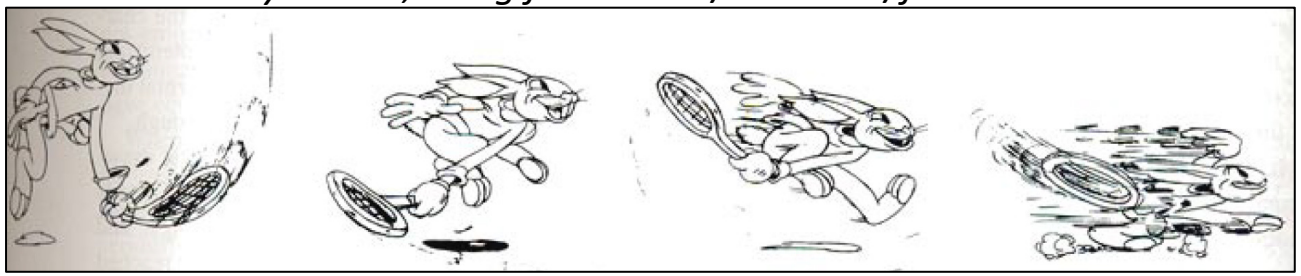
Often exaggerated deformations

Visible even on static images

Expressive animation:

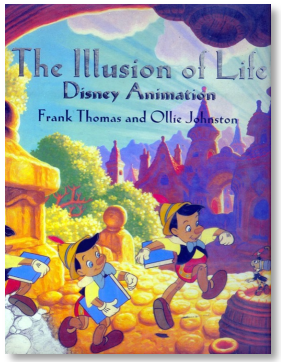
"Provide visual clues about the animation dynamics, using few details/elements/frames"

!= Realistic Animation

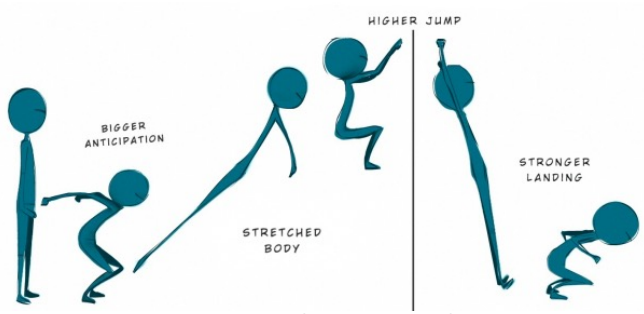


Fundamentals Principles of Animation (2D)

1. Timing
2. Spacing
3. **Slow-in, Slow-out**
4. **Squash & Stretch**
5. Anticipation
6. **Exaggeration**
7. **Follow Through**
8. **Secondary Action**
9. **Arcs**
10. Appeal
11. Staging
12. Straight Ahead/Pose to Pose



[F. Thomas and O. Johnston, Disney – The Illusion of Life, 1981]

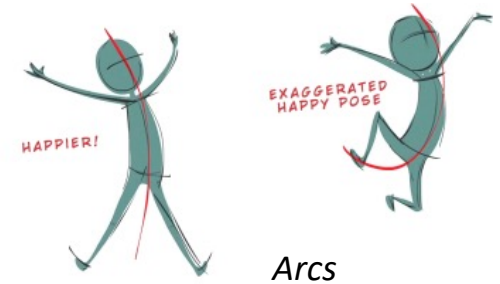


Squash & stretch



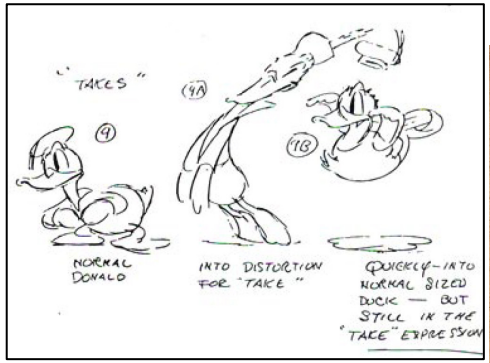
Follow Through

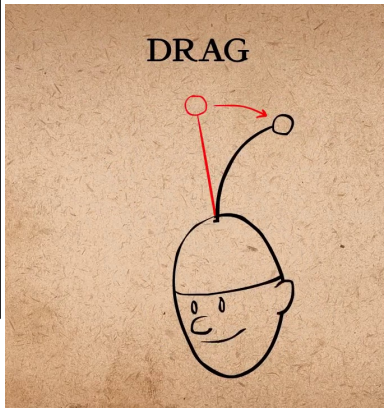


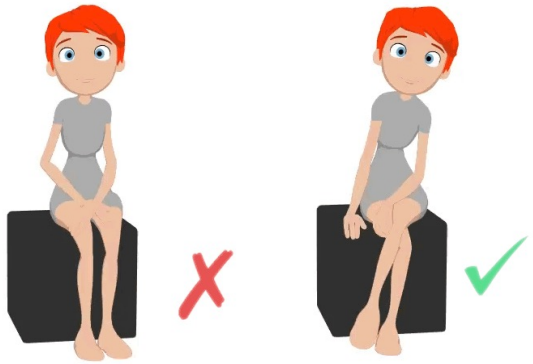


Arcs

Drag







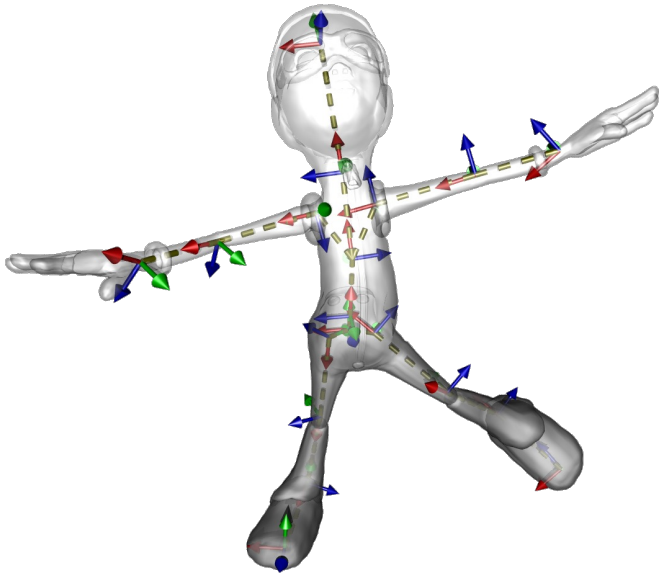
General objective of this work

1) Cartoon effects

Drag, Stretch, Followtrough, Oscillation, Delay

2) Automatically adapts to standard 3D Animated Character

Typical Skeleton-Based Skinning used in production



Constraints

Real time (>25 fps)

10^6 triangles

Work "Out of the box"

With user control/Art Directability

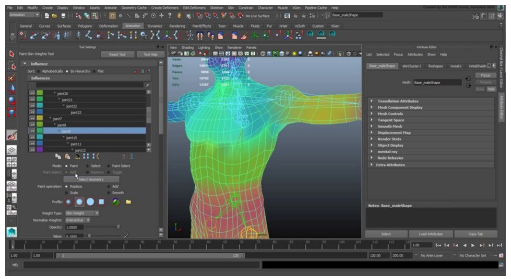


Overview

0. Background & Related Works

- **Standard Production Pipeline**
- **Linear Blend Skinning (LBS)**
- **Related Works**

Standard methods



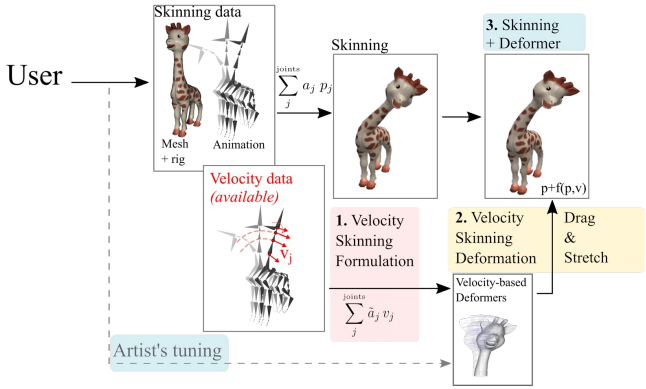
1. Hierarchy-based expression for LBS Velocity

2. Velocity Skinning Deformers (Drag, Stretch)

3. Acceleration Skinning (Follow-through)

4. Dynamic Skinning (Oscillation, Delay)

Contributions



Current Animation Production Pipeline

Pre-production

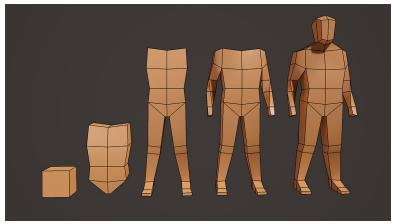
Storyboard, Animatic, Layout

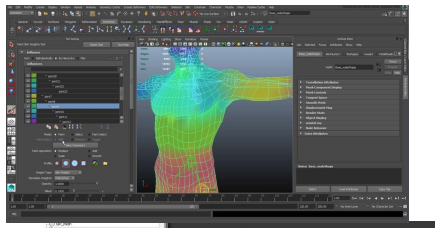
Production

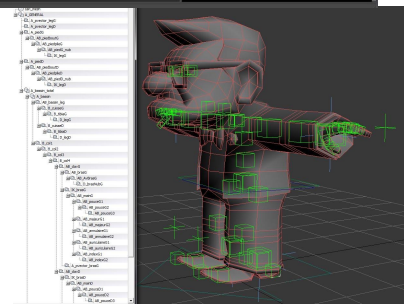
Modeling | Rigging | **Animation**

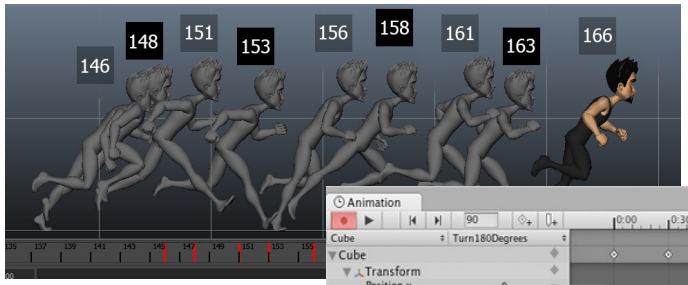
Post-Production

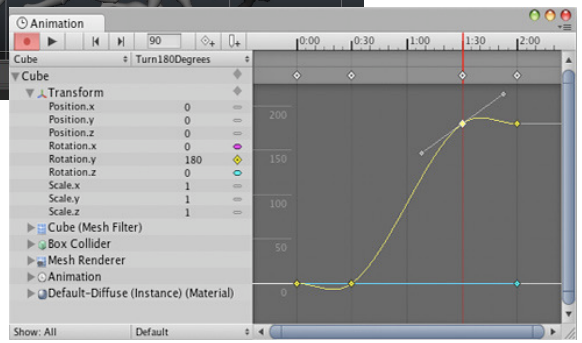
Rendering, Compositing, Editing











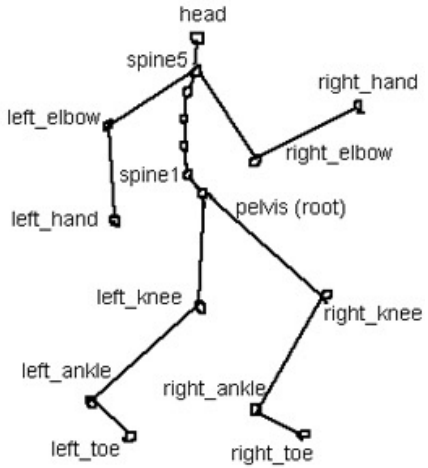
Skeleton based Character Animation

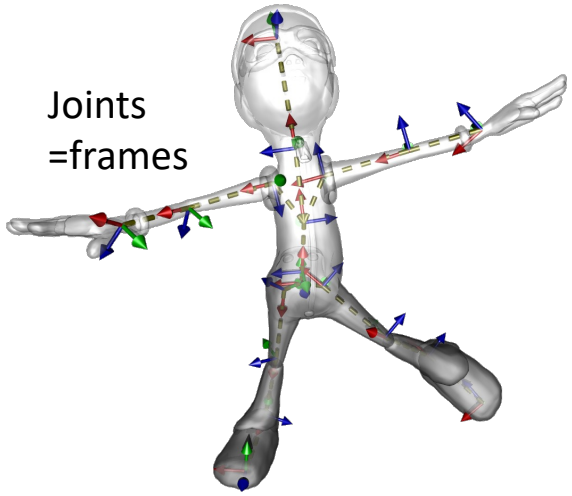
Skeleton = Set of joints organized in a hierarchy (tree, root=pelvis)

Joints = Frames, 4x4 matrix with Rotation & Translation

Degrees of freedom of the articulations

Bone = Segment between two joints



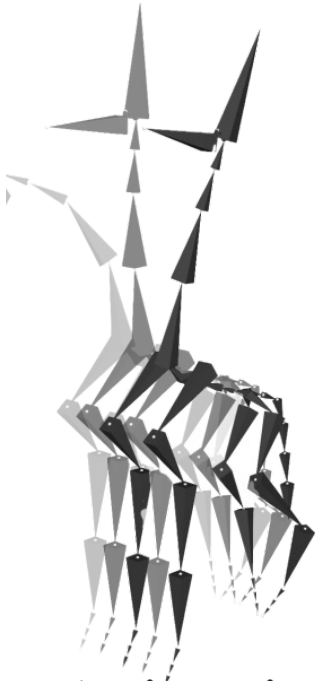


Animated skeleton: Rigid transformations applied to the joints

Mostly rotations around joint position

Joints are following **Rigid Motions**

T_j for joint j

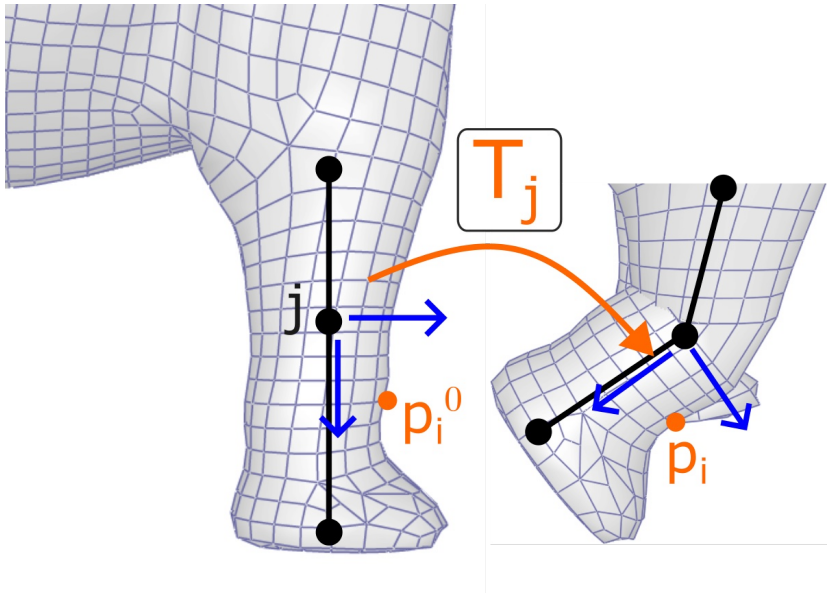


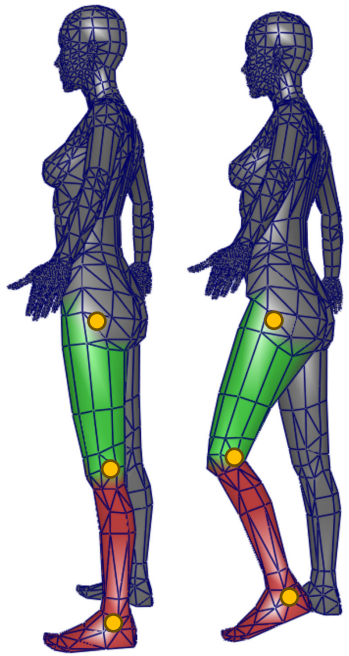
Rigid Skinning

Skinning = Deforming the “skin” surface on top of an articulated skeleton

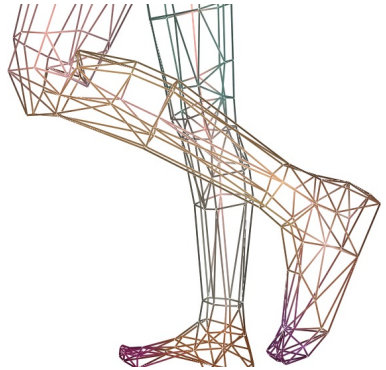
Rigid Skinning = 1 vertex attached to only 1 joint transformation

$$p_i = T_j \, p_i^0$$









(-) Discontinuities/self penetration at the joint level

Linear Blend Skinning (LBS)

1)
$$p_i = \sum_{joints\ j} \alpha_{ij}\ T_j\ p_i^0$$

α_{ij} – skinning weights
T_j – Joint transformation matrix

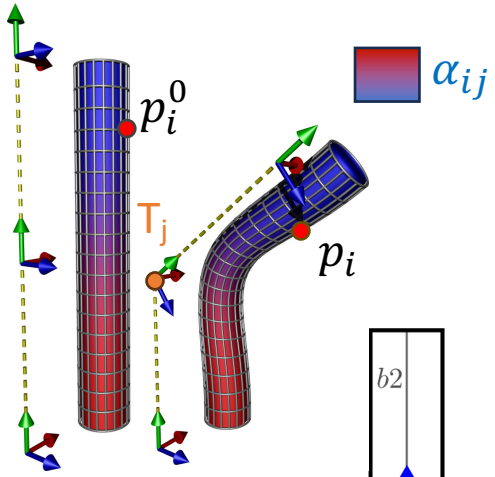
Linear blending between transformation matrices

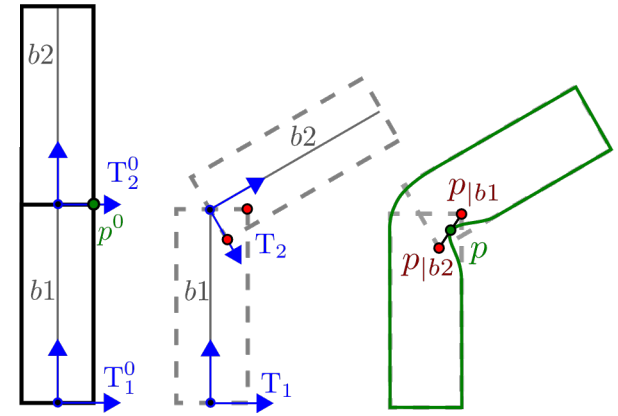
Similar formulation

2)
$$p_i = \sum_{joints\ j} \alpha_{ij}\ p_{i/j}$$

$p_{i/j}$ – positions from rigid skinning relative to joint j

Linear blending between positions from rigid skinning





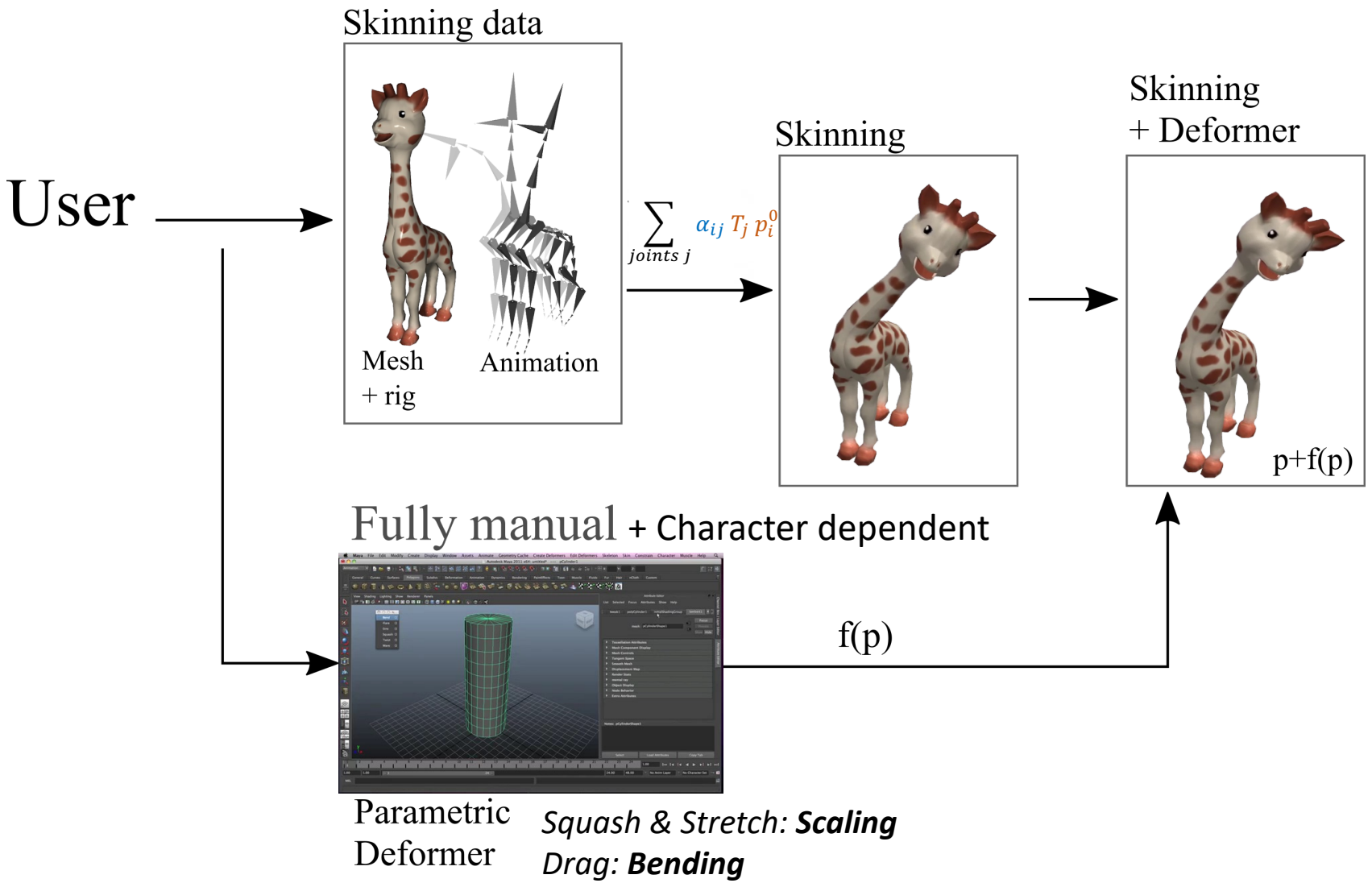


The most widely used method for character animation

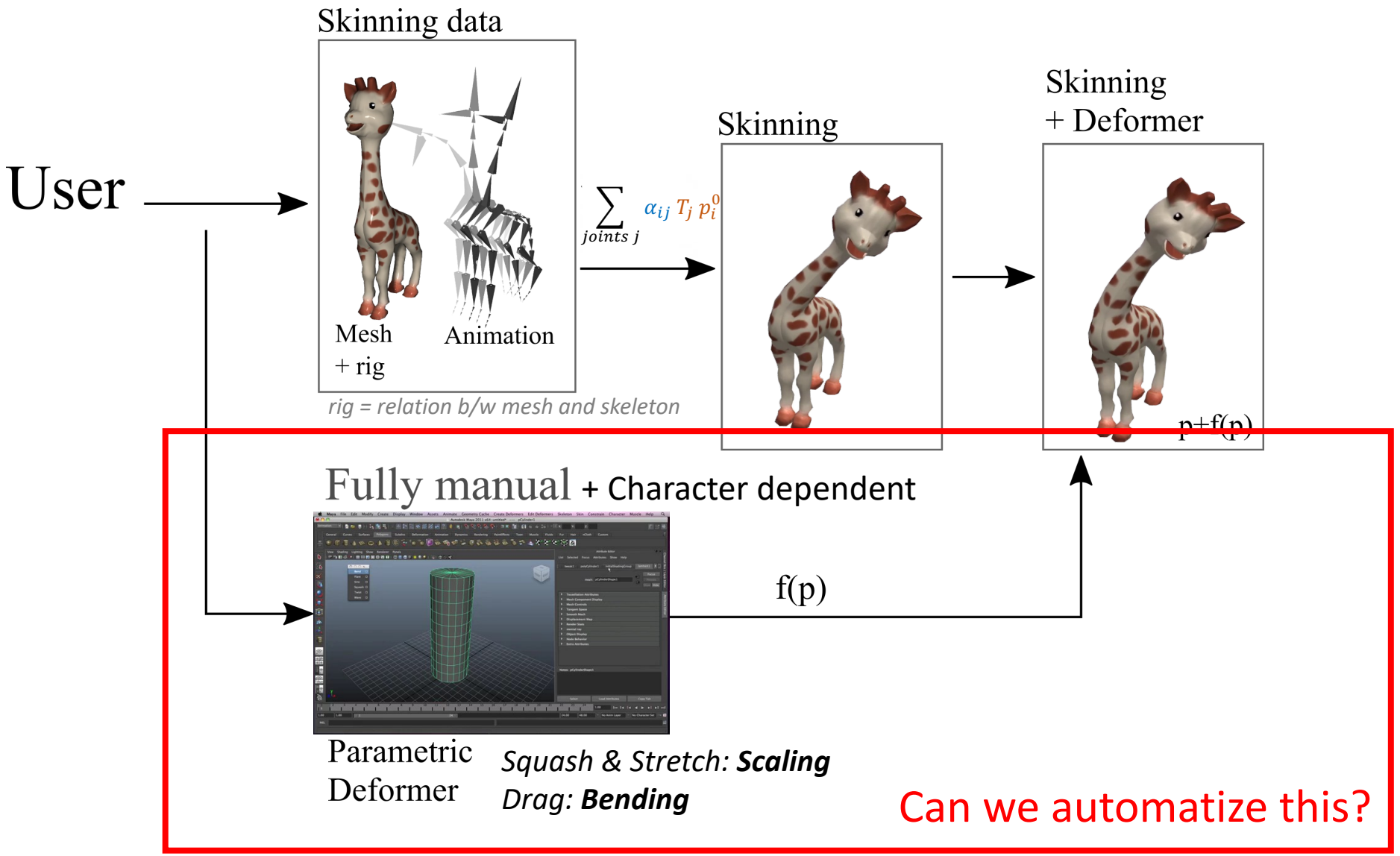
In video games, animation cinema, VFX, etc.

(-) Limited to static key-framed poses

Expressive animation in production



Expressive animation in production



Idea of our Work

2) Expressive deformation to global shape

Via parametric deformation
e.g. Squash & Stretch: ***Scaling***
Drag: ***Bending***



[John Lasseter – Principles of Traditional Animation Applied to 3D Computer Animation. SIGGRAPH 87]



[Noble & Tang, Automatic Expressive Deformations for Stylizing Motion*, CGF06]*

Simple & fast
But no hierarchy

1) Existing rigs available



$$p_i = \sum_{joints\ j} \alpha_{ij}\ T_j\ p_i^0$$

Notion of hierachy & transformation via joints and skinning weights

Simple & fast
But no expressive deformation

Can we **leverage existing rigs to generate expressive deformation** ?

Underlying idea:
Joint velocity: Magnitude of parametric deformers
Rig: Hierarchical propagation

Constraints*:*
- Real-time
- "As easy & direct as skinning" (no iterative simulation)
- Controlable



Related works – Expressive Character Animation

Animation

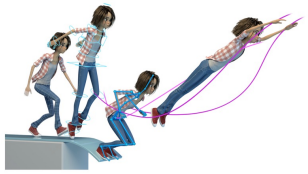
Manual

Automatic

Key Posing

Dynamic
Skinning

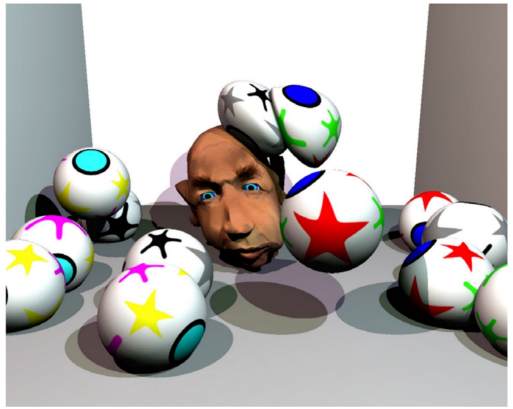
[Even EG23]
[Ciccone SCA17]
[Choi SIGGRAPH16
[Guay SIGGRAPH15]



Trajectory design

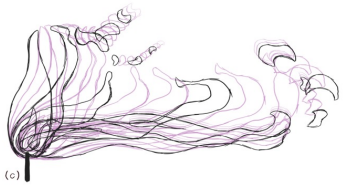


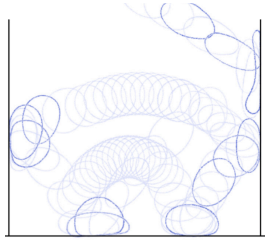
Physics



[Rumman14]
[Komaritzan18]
[Wu23]
[Xu16]
[Zhang20]
[Benchekroun23]
…

PPD
Reduced domains



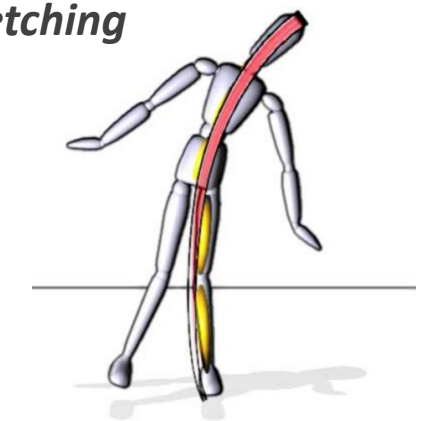


[Dvoroznak SIGGRAPH17]

[Hahn SCA15]

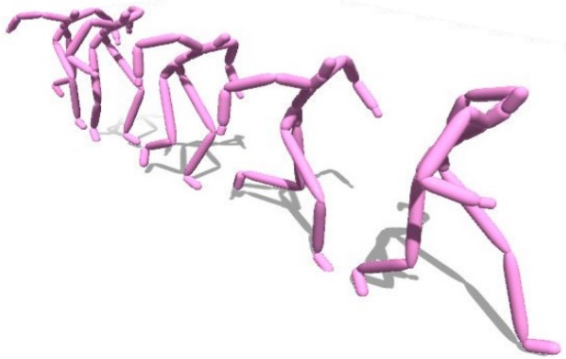
[Guay, SIGGRAPH Asia13]

Sketching





Pose transfert

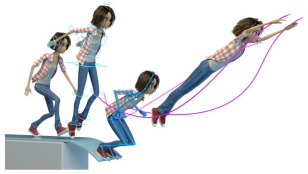


[Zhang 2023]
[Villegas 2018]
…

Related works – Expressive Character Animation

Animation

[Even EG23]
[Ciccone SCA17]
[Choi SIGGRAPH16
[Guay SIGGRAPH15]



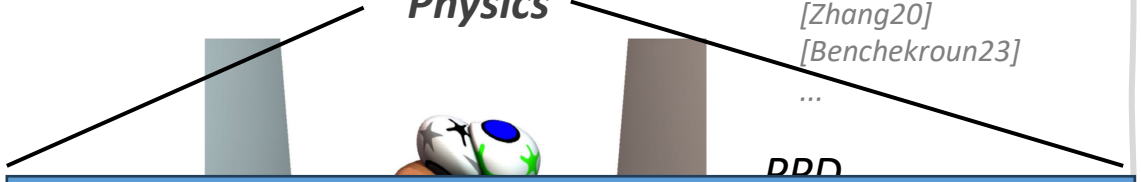
Trajectory design



Dynamic
Skinning

Physics

[Rumman14]
[Komaritzan18]
[Wu23]
[Xu16]
[Zhang20]
[Benchekroun23]
...

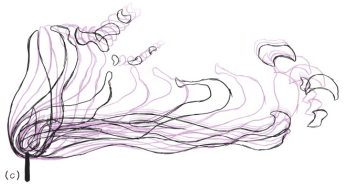


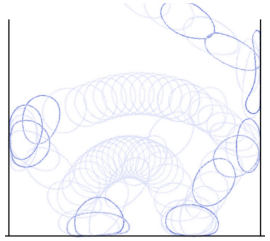
Physics-based approaches Limitations

(-) Numerical iterations:
- Divergence issues, computationally expensive.
- Move forward in time only (no “random access”).

(-) Difficult/limited parameter tuning:
- Indirect, non linear dependence to stiffnesses, Young modulus, etc.

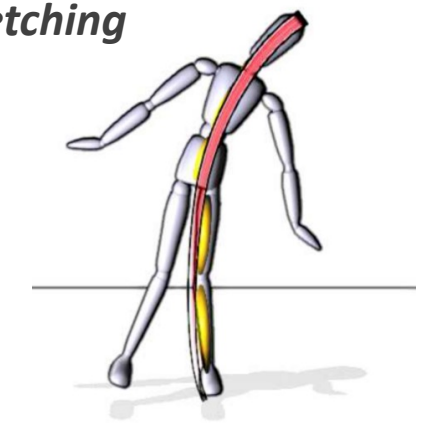
Manual





[Dvoroznak SIGGRAPH17]

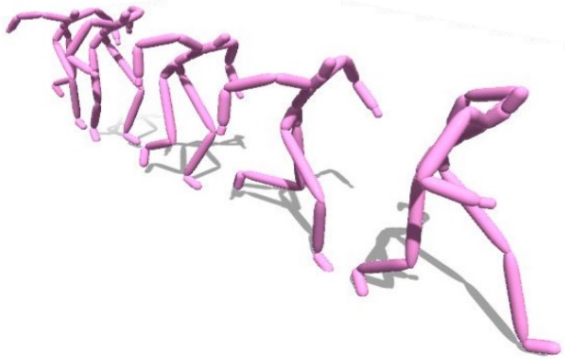
Sketching



[Hahn SCA15]
[Guay, SIGGRAPH Asia13]



[Villegas 2018]
...



Key Posing

Outline

0. Background & Related Works

1. Hierarchy-based expression for LBS Velocity

2. Velocity Skinning Deformers (Drag, Stretch)

3. Acceleration Skinning (Follow-through)

4. Dynamic Skinning (Oscillation, Delay)

What we would like to do

Dynamic skinning = LBS position (static) + Deformer φ depending on velocity (dynamic)

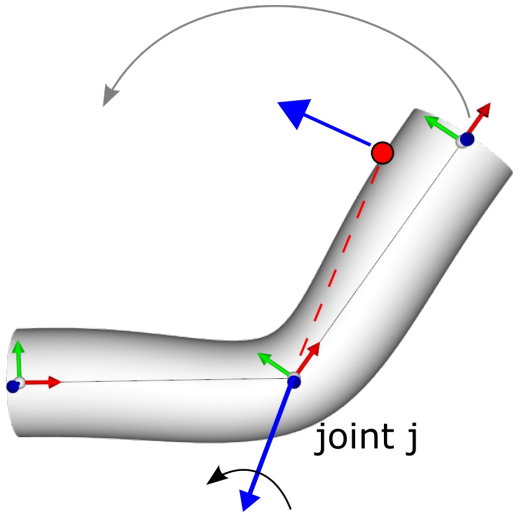
$$p_i^{DS} = p_i^{LBS} + \varphi(\dot{p}_i)$$

Problem: Defininig φ in general is complicated

But: Defining φ for a single joint motion is simple

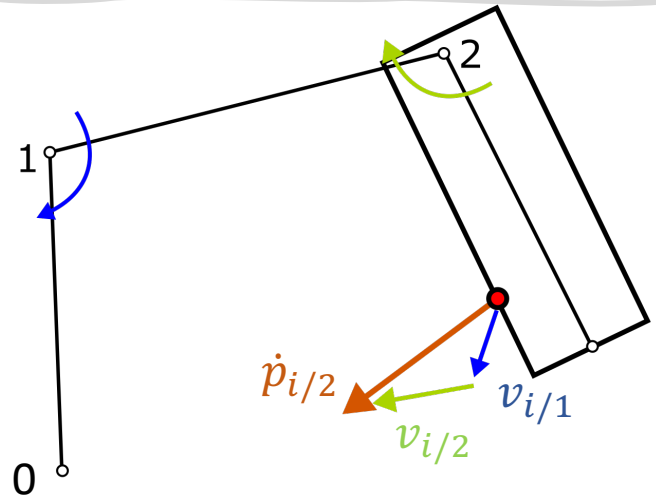
Joint => Rigid Motion => Simple deformer (scaling, bending, etc)

Idea: Use Linear Blend Skinning to Blend multiple simple φ for each joint



Objective: Express the vertex velocity $\dot{p}_i$ as combination of joint-based rigid motion velocities

Hierarchy-based expression for LBS Velocity



Standard LBS $p_i = \sum_{joints\ j} \alpha_{ij}\, p_{i/j}$

Time derivative $v(p_i) = \dot{p}_i = \sum_{joints\ j} \alpha_{ij}\, \dot{p}_{i/j}$

$\dot{p}_{i/j} = \sum_{j_a \in Anc(j)} v_{i/j_a}$

= velocity induced by kinematic chain up to joint j

= sum of all individual joint velocity

Double summation … $\dot{p}_i = \sum_{joints\ j} \alpha_{ij} \left(\sum_{j_a \in Anc(j)} v_{i/j_a} \right)$

Can be re-arranged: $\boxed{\dot{p}_i = \sum_{joints\ j} \beta_{ij}\ v_{i/j}}$

$\beta_{ij} = \sum_{k \in Desc(j)} \alpha_{ik}$

Hierarchical weights

Hierarchy-based expression for LBS Velocity

Standard LBS

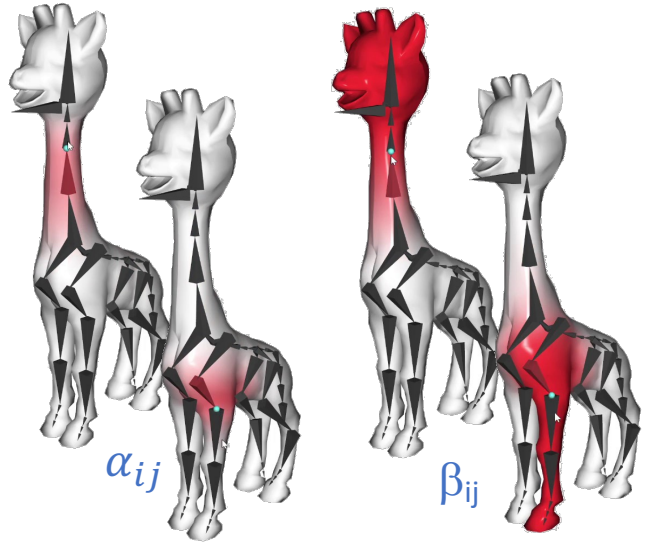
$$p_i = \sum_{joints\ j} \alpha_{ij}\, p_{i/j}$$

LBS-like Velocity

$$\dot{p}_i = \sum_{joints\ j} \beta_{ij}\ v_{i/j}$$

$$\beta_{ij} = \sum_{k \in Desc(j)} \alpha_{ik}$$

Hierarchical weights

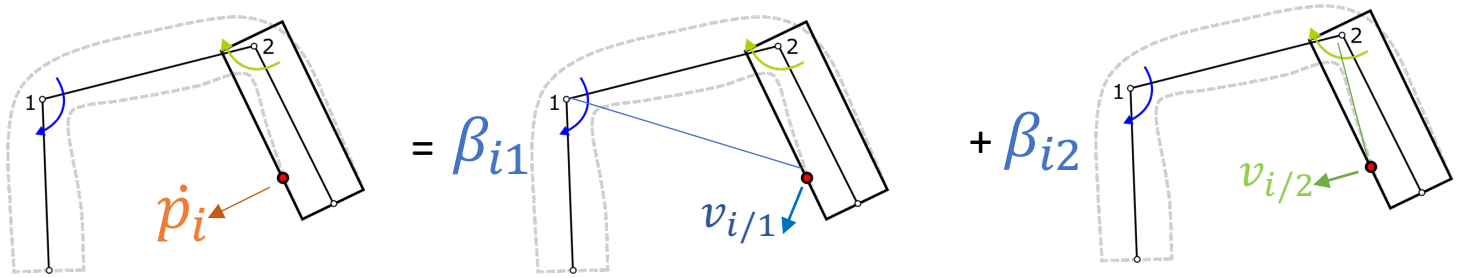


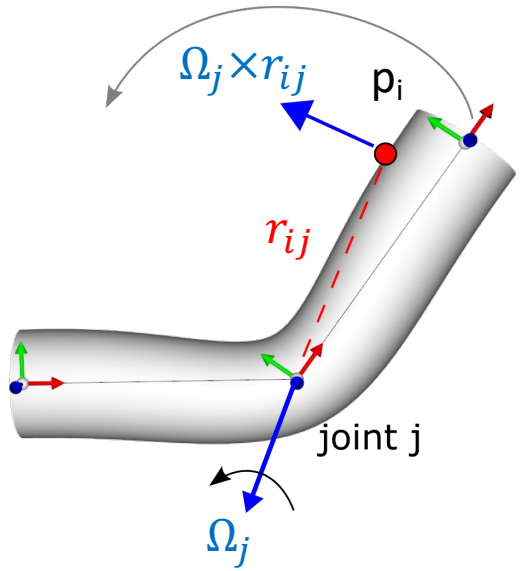
Vertex velocity = **linear combination** of **rigid-motion** velocities

$$v_{i/j} = \underbrace{L_j + \Omega_j \times r_{ij}}_{\text{Depends on joint parameters}}$$

L: Linear velocity
Ω: Angular velocity
r_{ij}: vertex to joint/pivot

Skeleton **hierarchy** expressed bia β_{ij}





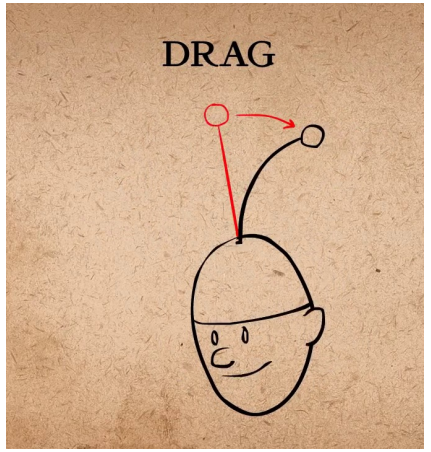
Outline

1. Hierarchy-based expression for LBS Velocity

2. Velocity Skinning Deformers (Drag, Stretch)

3. Acceleration Skinning (Follow-through)

4. Dynamic Skinning (Oscillation, Delay)

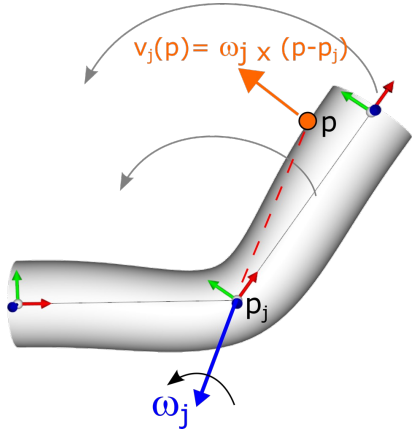




Velocity Skinning General Formulation

Velocity Skinning = LBS position + Deformer φ depending on velocity

$$p_i^{VS} = p_i^{LBS} + \varphi(\dot{p}_i) \quad \text{where} \quad \dot{p}_i = \sum_{joints\ j} \beta_{ij}\ v_{i/j}$$



Split φ into many procedural deformers, and blend them

$$p_i^{VS} = p_i^{LBS} + \sum_{joints\ j} \beta_{ij}\ \varphi(v_{i/j}) \quad \text{where} \quad v_{i/j} = L_j + \Omega_j \times r_{ij}$$

$$p_i^{VS} = p_i^{LBS} + \sum_{joints\ j} \beta_{ij}\ (\varphi^{Lin}(L_j) + \varphi^{Ang}(\Omega_j, r_{ij}))$$

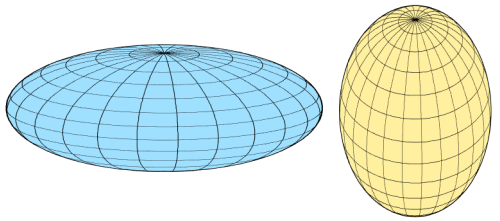
- A deformation for each moving joint
- Split b/w linear and angular velocity



Automatic weighting along hierachy

Origin of the motion

Can define φ as procedural deformer, in joint space

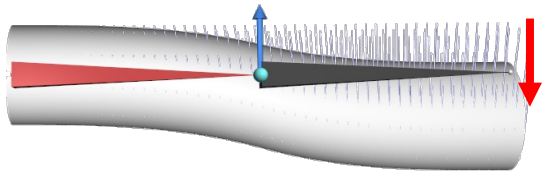


Velocity Skinning Deformation: Drag Effect

Drag = Deformation opposite to the ongoing motion of the joint

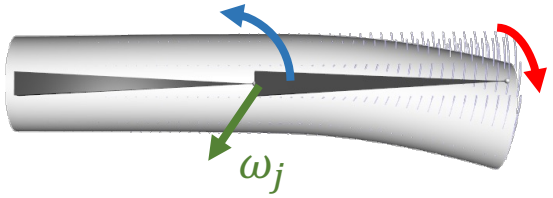
Linear velocity:
Translation *opposite to joint velocity*

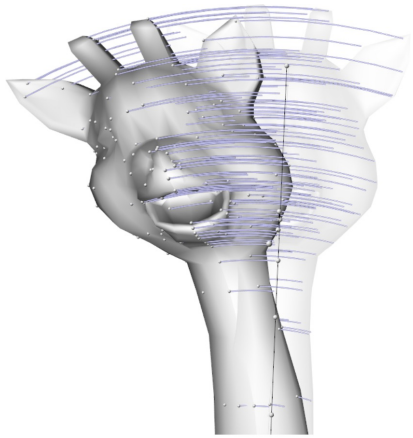
$$\varphi^{Lin} = -K^{Lin}\, L_j$$

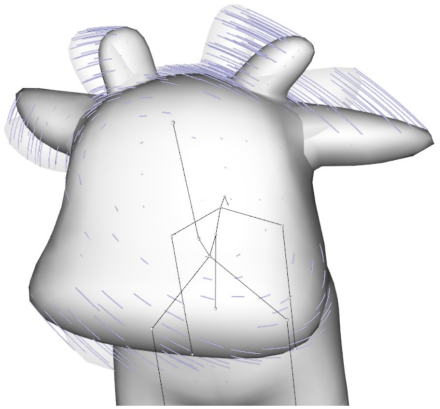


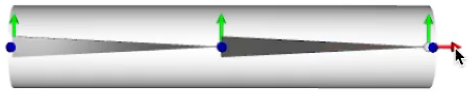
Angular velocity:
Rotation *opposite to joint angular velocity* Ω_j

$$\varphi^{Angl} = Rot\left(K^{Ang}||\Omega_j \times r_{ij}||, \Omega_j\right)$$





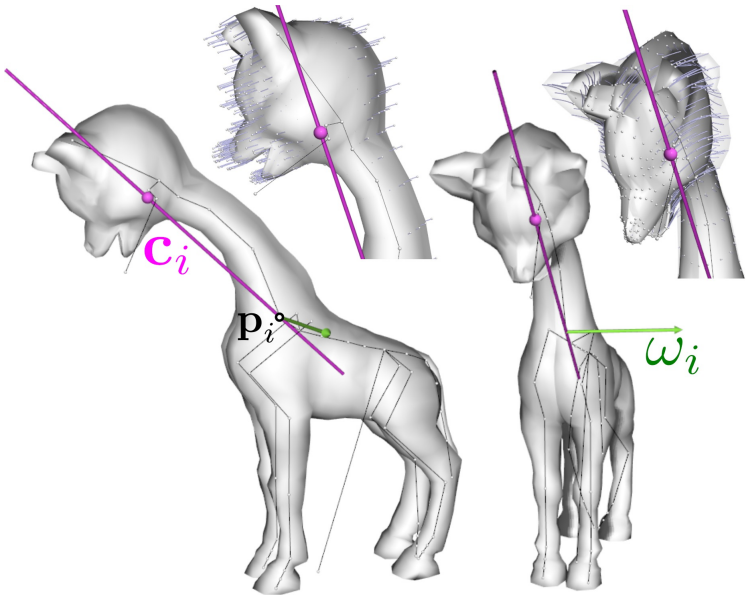






Velocity Skinning Deformation: Stretch Effect

1) Define a local frame for each joint
p_j, Ω_j, c_j (center of mass of influenced vertices)



2) Stretch effect as scaling in this joint frame

Linear velocity

$\psi_{u_j}^{stretch}$ Homogeneous Scaling



Angular velocity

$\psi_{u_j}^{stretch}$ Anisotropic Scaling



Results



Original Mesh



Standard Skinning



Our Results
Velocity Skinning

Replicability Stamp Initiative

```
Algorithm:

 Precomputation:
  Compute ã from a
  Compute initial centroid cj for each joint

 Run Time:
   For all vertices at position p:
     For all joint j at position pj:
       (uj,wj) <- (linear,angular) velocity of j
       d_drag    = drag_linear(uj)          + drag_linear(wj, p, pj)
       d_stretch = stretch_linear(uj, cj)   + stretch_angular(wj, p, pj, cj)
       d = d_drag + d_stretch
       p = p + ã  * d
```

https://velocityskinning.com

Code (C++/Shader) + Interactive JS on github

[Rohmer, Tarini, Kalyanasudaram, Moshfeghidar, Cani, Zordan. Velocity Skinning for Real-time Stylized Skeletal Animation, Eurographics 2021]

Results

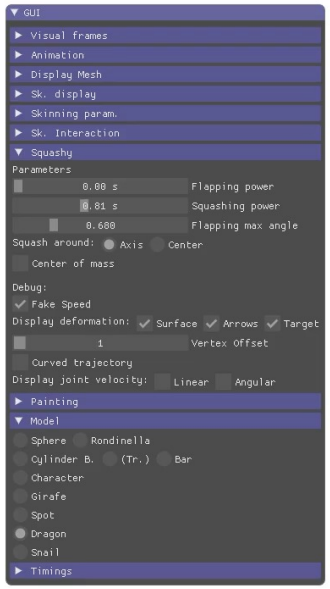


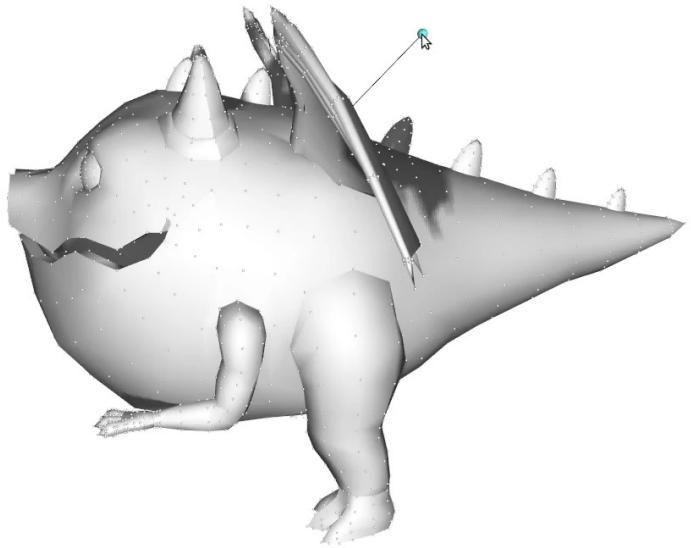
Standard Skinning



Velocity Skinning

Results – User controls





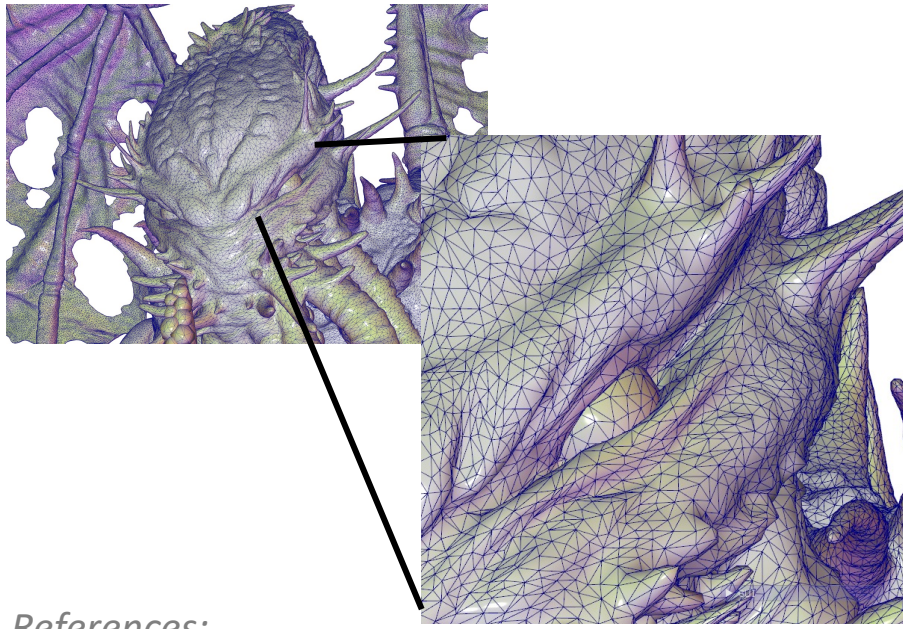
Deformation Visualizer

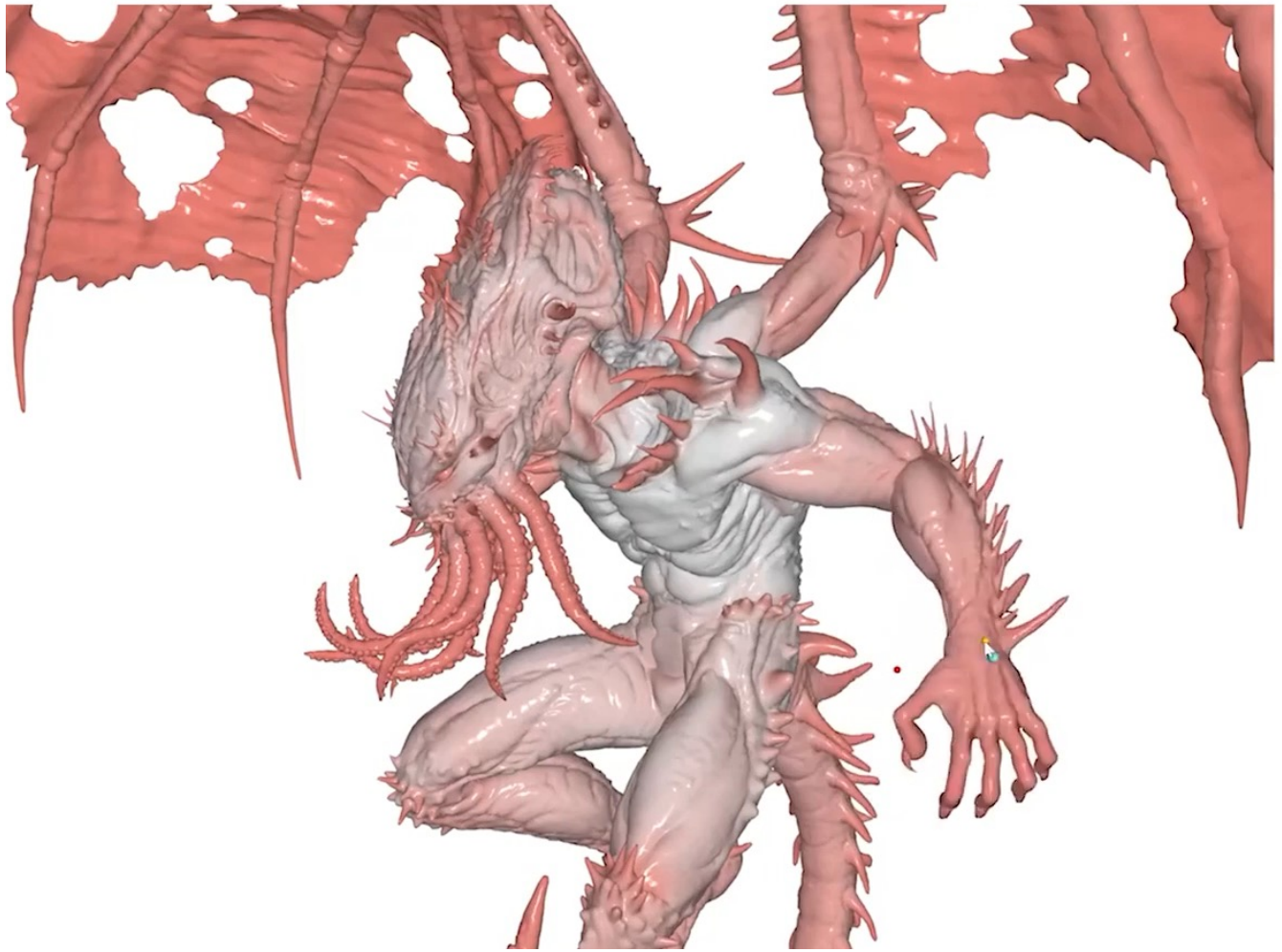
Results – Computational Time

Large mesh

1.5 Million triangles
< 37ms/frame

Single pass vertex shader



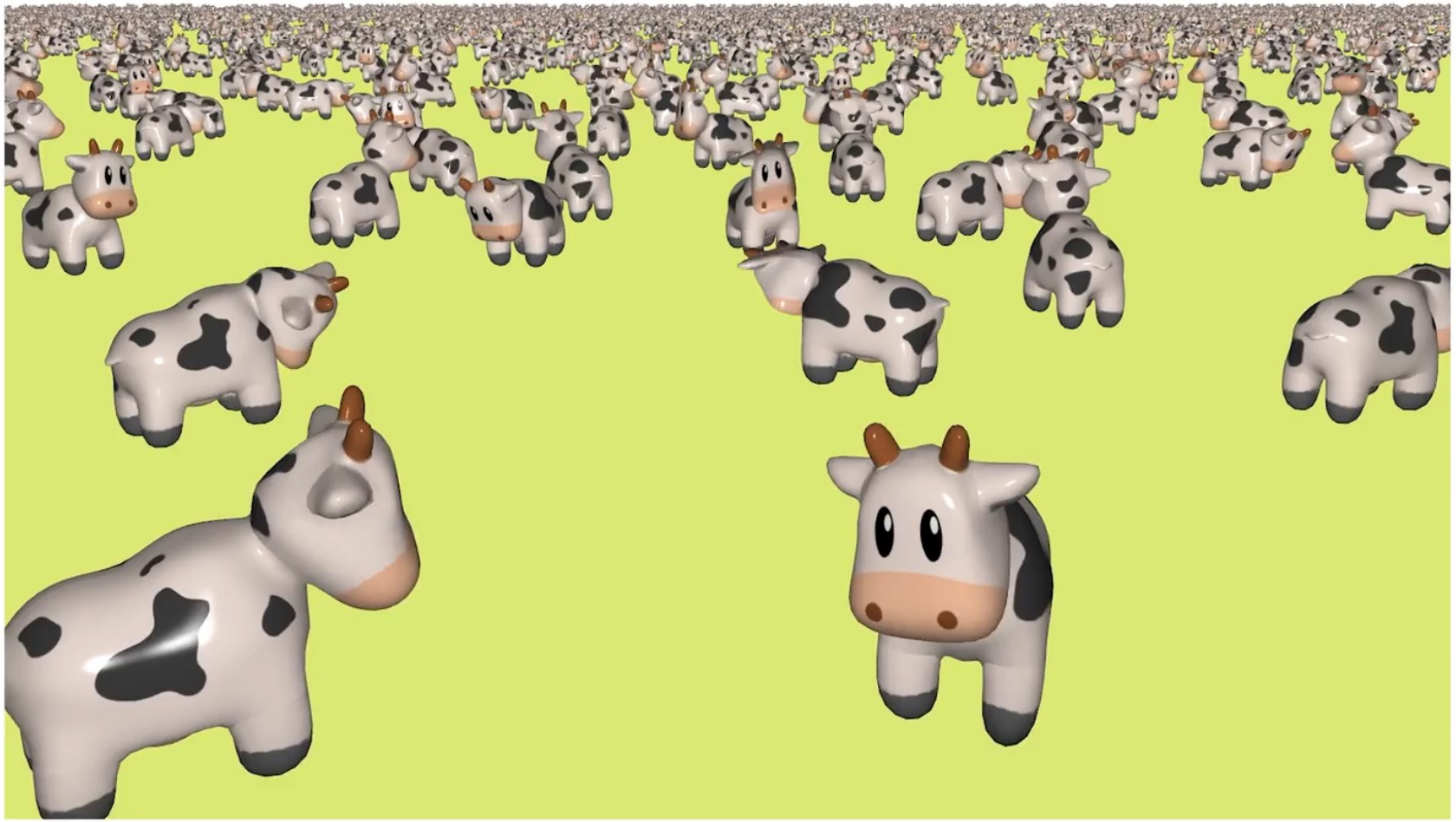


References:
[Benchejroun et al., Fast Complementary Dynamics, SIGGRAPH23] 14s/frame for 0.3 Million triangles (x2000).
[Wu, Umetani, Two-Way coupling Skinning and PBD, SIGGRAPH23] 17ms/frame for 4200 triangles.

Results – Computational Time

Multiple instances

4000 cows,
5800 triangles/cow
<40 ms/frame

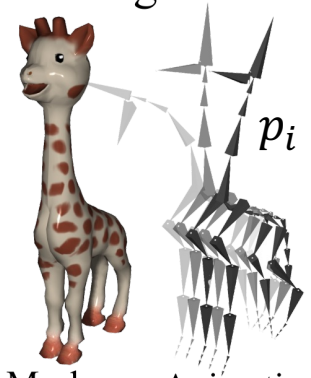


Velocity Skinning - Summary

$$p_i^{VS} = p_i^{LBS} + \sum_{joints\ j} \beta_{ij}\ \varphi(v_{i/j})$$

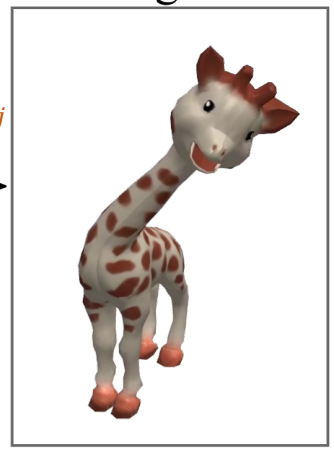
User

Skinning data



$$\sum_{joints\ j} \alpha_{ij}\ p_{i/j}$$

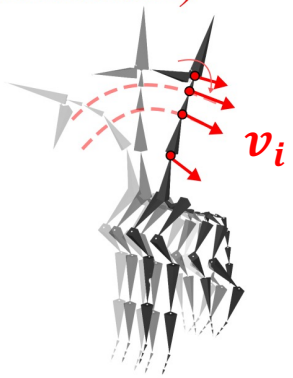
Skinning



3. Skinning
+ Deformer



Velocity data
(available)

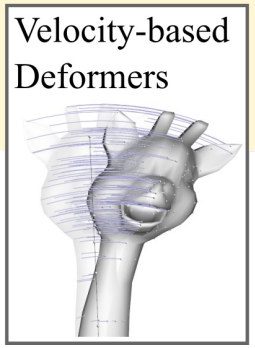


1. Velocity
Skinning
Formulation

$$\dot{p}_i = \sum_{joints\ j} \beta_{ij}\ v_{i/j}$$

2. Velocity
Skinning
Deformation

Velocity-based
Deformers



Drag,
Stretch

$$\sum_{joints\ j} \beta_{ij}\ \varphi(v_{i/j})$$

Artist's tuning

Outline

1. Hierarchy-based expression for LBS Velocity

2. Velocity Skinning Deformers (Drag, Stretch)

3. Acceleration Skinning (Follow-through)

4. Dynamic Skinning (Oscillation, Delay)



Acceleration Skinning Principles

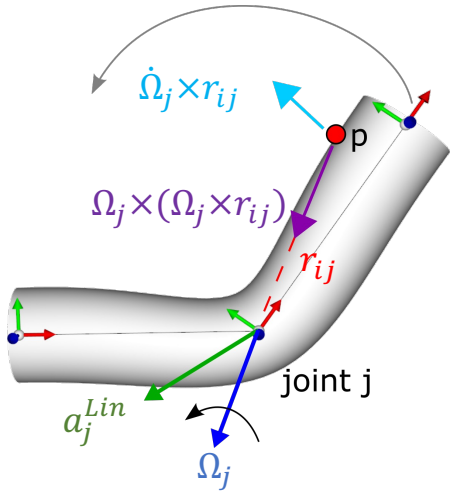
Hierarchical relation also valid for acceleration

$$v_i = \sum_{joints\ j} \beta_{ij}\, \dot{p}_{i/j} \longrightarrow a_i = \dot{v}_i = \sum_{joints\ j} \beta_{ij}\, \ddot{p}_{i/j}$$

Decomposition of rigid transform acceleration with joint parameter:

$$\ddot{p}_{i/j} = a_j^{Lin} + \dot{\Omega}_j \times r_{ij} + \Omega_j \times (\Omega_j \times r_{ij})$$

Linear acceleration — *Angular acceleration* — *Centripetal component*



Deformation parameterized with acceleration quantities:

$$p_i^{AS} = p_i^{LBS} + \sum_{joints\ j} \beta_{ij}\, \varphi(\boldsymbol{a_j^{Lin}}, \boldsymbol{\dot{\Omega}_j}, \Omega_j, r_{ij})$$

Acceleration Skinning Effects: Follow Through

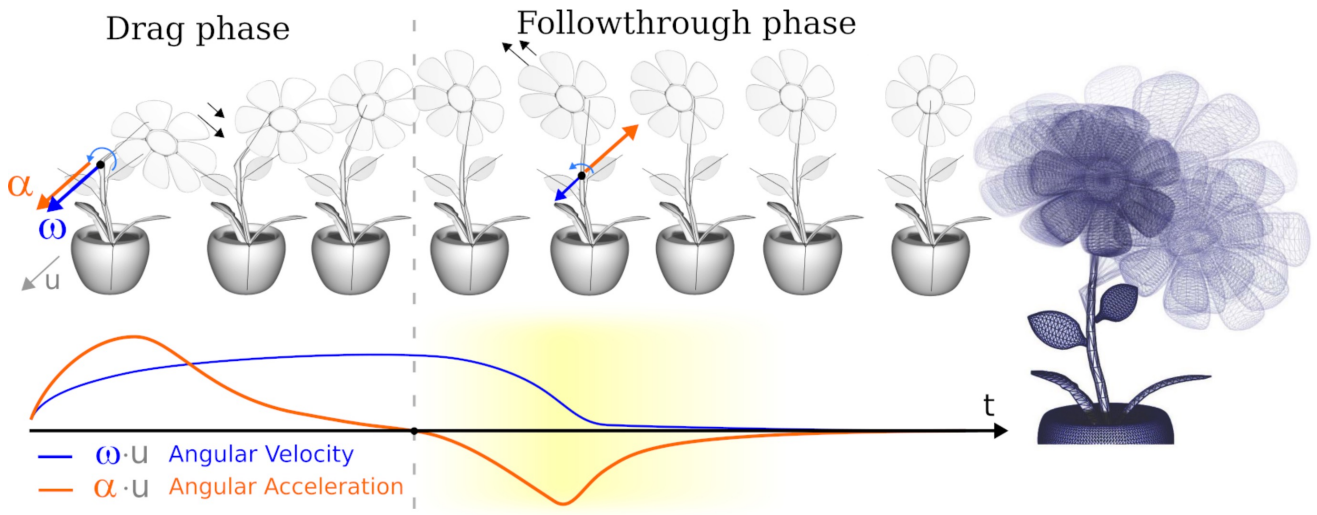
Detector: Acceleration VS Deceleration $[\Omega_j \cdot \dot{\Omega}_j] = \text{clamp}(\Omega_j \cdot \dot{\Omega}_j / K, 0, 1)$

Same parametric deformer

Acceleration instead of velocity

Follow through Angular $\psi^{FT} = [\Omega_j \cdot \dot{\Omega}_j] \ \psi^{drag}(\dot{\Omega}_j, r_{ij})$

Follow through Linear $\psi^{FT} = [v_j^{Lin} \cdot a_j^{Lin}] \ \psi^{drag}(a_j^{Lin})$





Acceleration Skinning Effects: Results

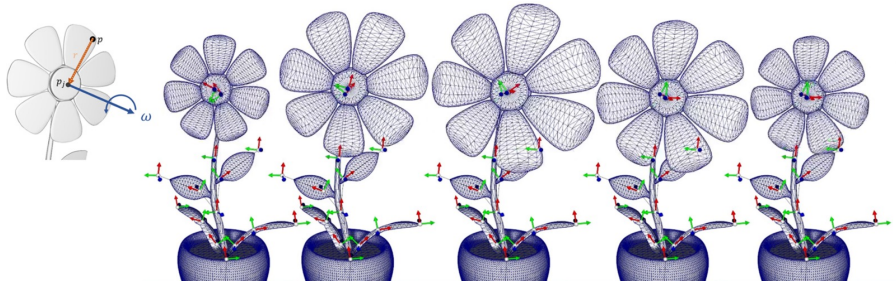


[Kalyanasundaraman et al., Acceleration Skinning: Kinematics-Driven Cartoon Effects for Articulated Characters. Graphics Interface, 2022]

Acceleration Skinning Effects: Centrifugal

Centrifulag Stretch

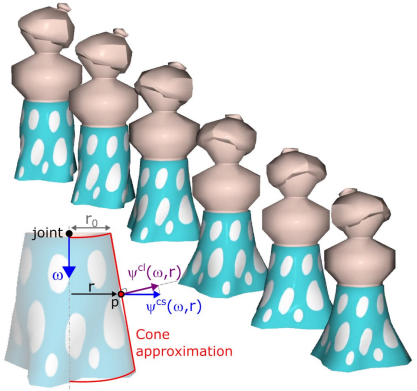
$$\varphi^{CStretch} = -K\ \Omega_j \times (\Omega_j \times r_{ij})$$





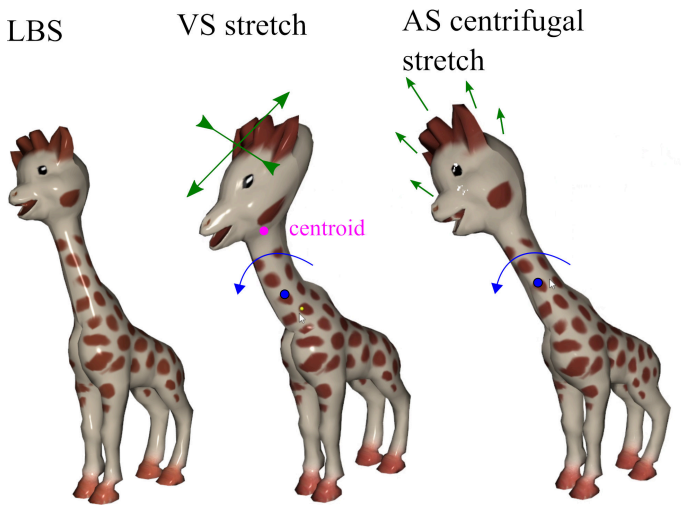
Centrifugal Lift: Constrained stretch in normal direction

$$\varphi^{CLift} = \chi(\text{rij})\ (\psi^{CStretch}(\omega_j, r_{ij}) \cdot n)\ n$$





Acceleration Skinning: Discussion



(+) As simple & efficient that velocity
(+) Closer to physics & force model
(+) Allows extra effect
- Centripetal accel. -> (+) Well defined stretch effect
- Follow through

(-) Acceleration sensitive to noise
(-) Harder to parameterize
(-) Follow through -> single overshoot

=> Idea: Extend velocity-based deformation to oscillating behavior using time-filter

Outline

1. Hierarchy-based expression for LBS Velocity

2. Velocity Skinning Deformers (Drag, Stretch)

3. Acceleration Skinning (Follow-through)

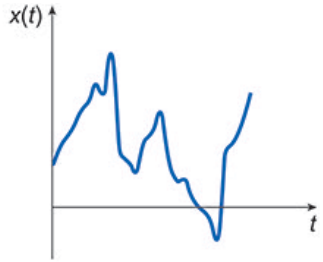
4. Dynamic Skinning (Oscillation, Delay)

(ongoing work)

Dynamic Skinning: Principle

Add time-filtered signal layer

$$p_i^{VS} = p_i^{LBS} + \sum_j \beta_{ij}\, \varphi(\Omega_j, r_{ij})$$



$$\Omega_j \rightarrow F(\Omega_j, t)$$

$$p_i^{DS}(t) = p_i^{LBS}(t) + \sum_j \beta_{ij}\, \varphi\big(F(\Omega_j, t), r_{ij}\big)$$

F(. , t) : Linear causal time filter

=> How to define F ?

Oscillating Behavior, Filter F_o

$$p_i^{DS}(t) = p_i^{LBS} + \sum_j \beta_{ij}\, \psi\left(F_o\left(\Omega_j, t\right), r_{ij}\right)$$

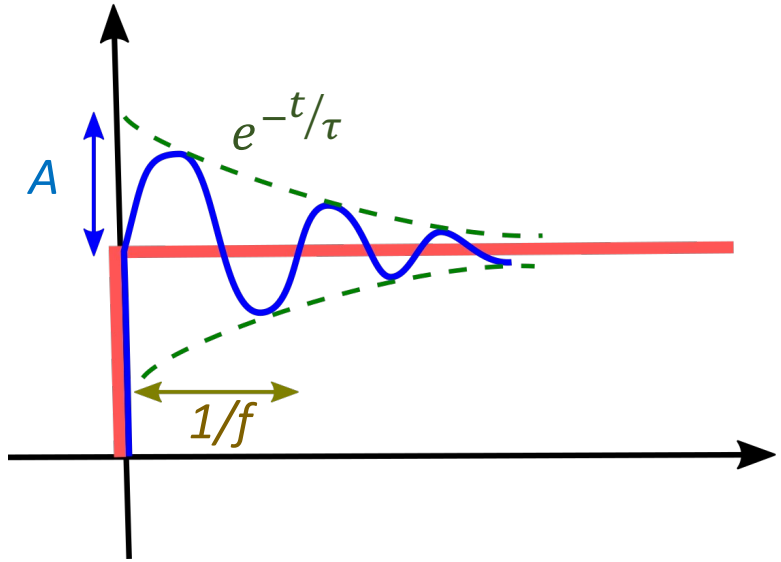
Define F_o as response to "unit step" U(t)

$$h(t) = \left(1 + \kappa(t)\right) U(t)$$

$$\kappa(t) = A \exp(-t/\tau) \sin(2\pi f\ t)$$

A: Magnitude
τ: damping
f: frequency

Explicit control
Independent parameters



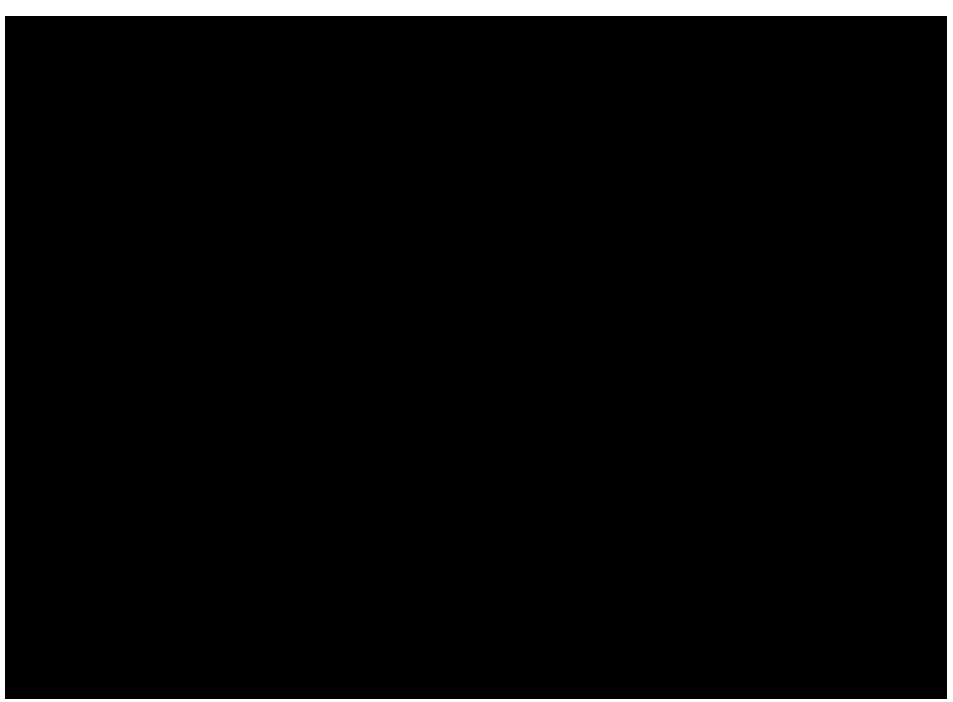
Expression of filtered signal:

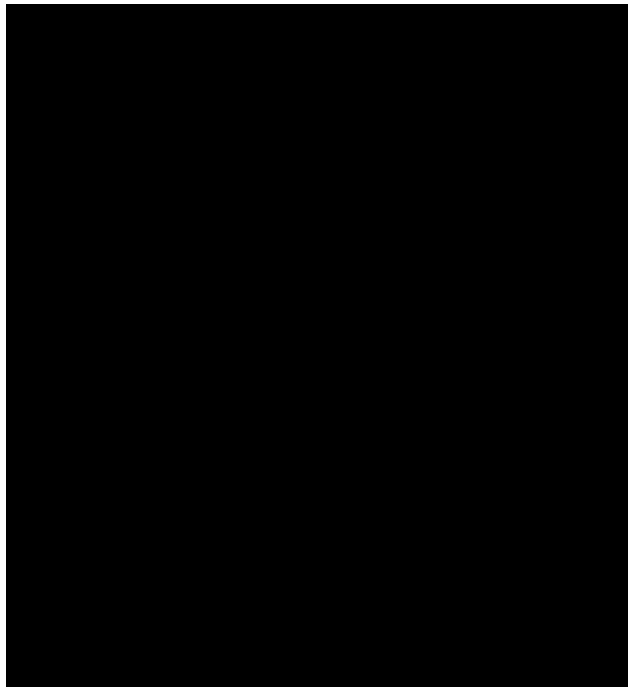
Convolution with "impulse response" = derivative of "unit step response"

$$F_o\left(\Omega_j, t\right) = (\dot{h} * \Omega_j)(t)$$

$$F_o\left(\Omega_j, t\right) = \Omega_j(t) + \int_u \dot{\kappa}(u)\, \Omega_j(t-u)\, du$$

Oscillating Behavior - Results



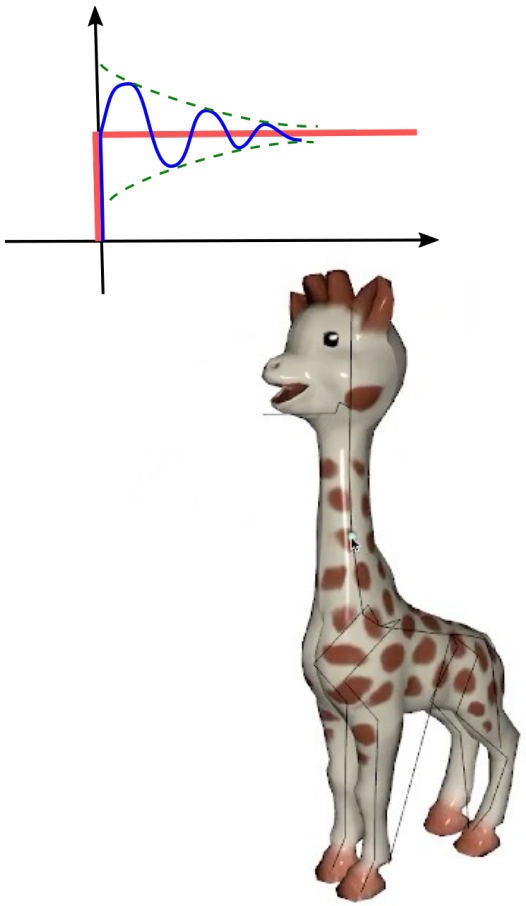




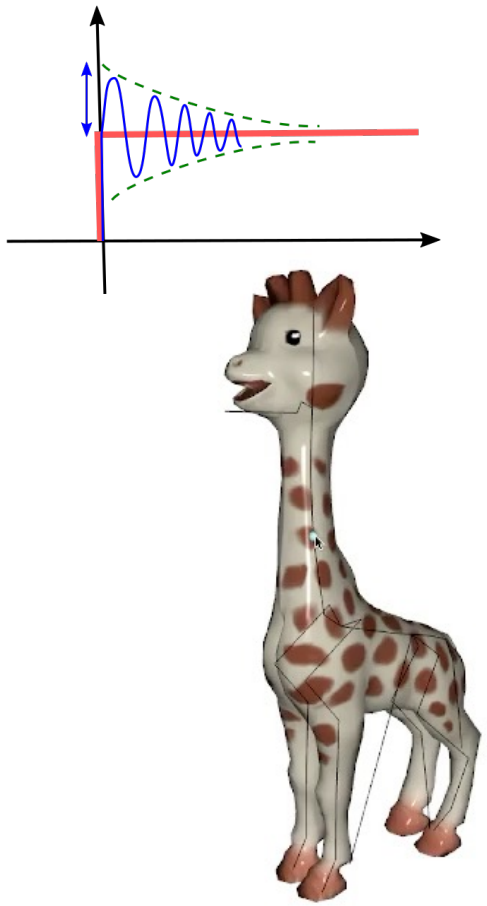
(+) Simple to parameterize
(+) Long term oscillation

(-) Need the past skeleton state
(-) A bit more costly (convolution)

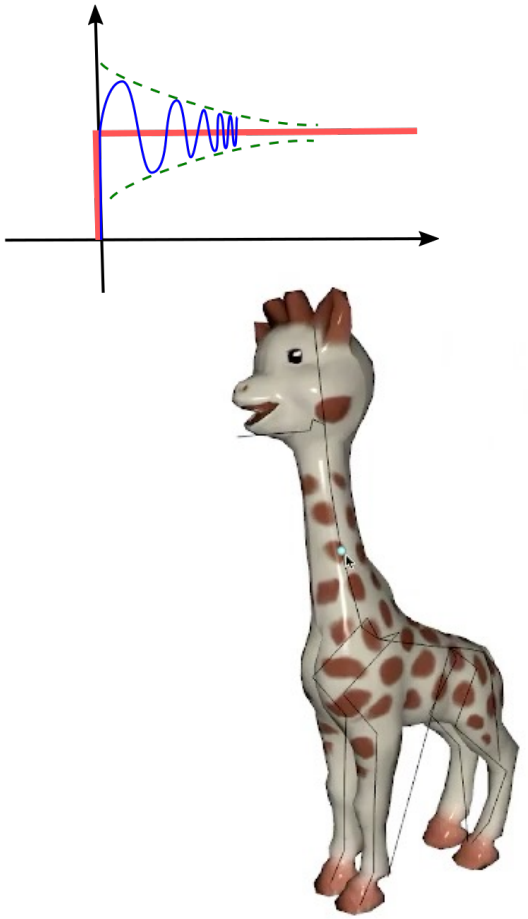
Oscillating Behavior - Results



Base frequency



Higher frequency



Increasing frequency

Extension to delayed Skinning: dLBS

Apply a time-filter on the LBS itself
Time delay on transformation matrices

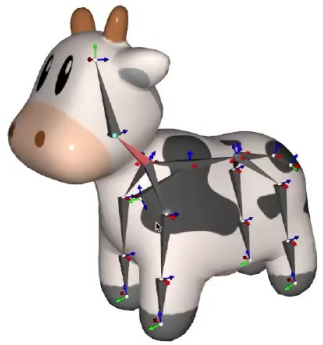
$$p_i^{LBS}(t) = \sum_j \alpha_{ij}\ T_j(t)\ p_i^0$$

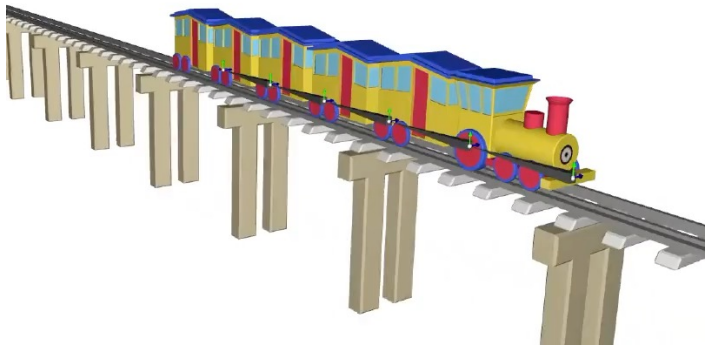
$$p_i^{dLBS}(t) = \sum_j \alpha_{ij}\ F_d(T_j, t)\ p_i^0$$

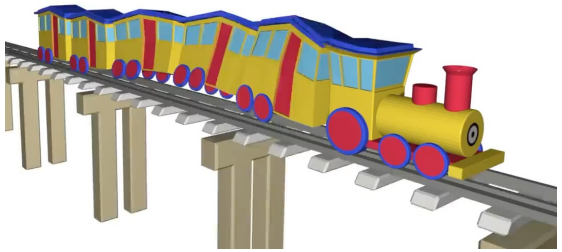
$$F_d(T_{j_1}, t) = \prod_{j \in Anc(j_1)} T_j^{Loc}(t - d_{ij}/c) = T_0^{Loc}(t - d_{i0}/c) \cdots T_{j_1}^{Loc}(t - d_{ij_1}/c)$$

Wave propagation F(t-distance/speed)

dij: distance vertex i to joint j
c: wave propagation speed







Conclusion

General framework “similar to LBS” in velocity

(+) Express procedural deformer with joint kinematics

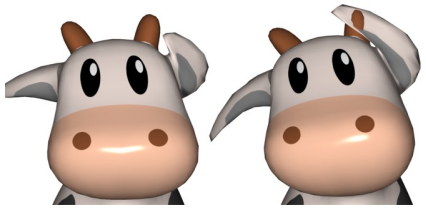
(+) Simple method, robust and very fast

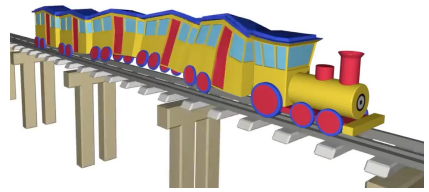
(+) Effects: Drag, Stretch, Follow-trough, Oscillation, Delay

$$\dot{p}_i = \sum_{joints\ j} \beta_{ij}\ v_{i/j}$$

$$p_i^{DS}(t) = p_i^{LBS}(t) + \sum_{joints\ j} \beta_{ij}\ \varphi(v_{i/j}, t)$$







Limitation: Collision, linear blending

Discussion: **Deformation propagation guided by skeleton hierarchy**

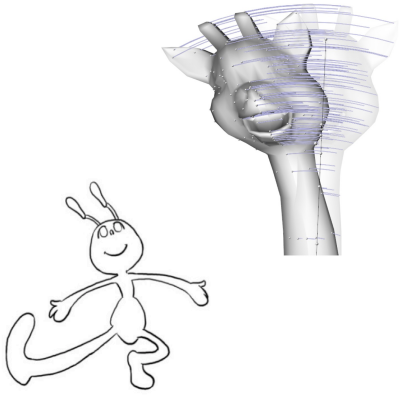
(+) Rely on existing rigged character

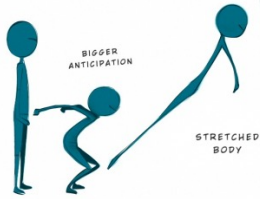
(-) Imposes propagation from root -> extremity

Not appropriate: Walking / impact response

Possible extensions

- Dynamic “virtual” skeleton adaptation (/impact)
- Extended effects: Anticipation, speed vibrations, environment, etc.
- Expressive deformer design, parameter fitting





[redacted]

[redacted]

[redacted]

Thank You

[redacted]

[redacted]



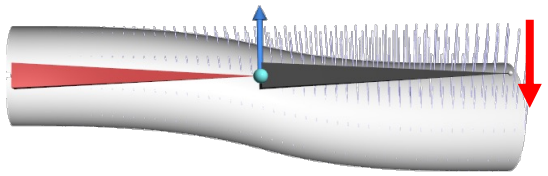
https://velocityskinning.com

Velocity Skinning Deformation: Drag Effect

Drag = Deformation opposite to the ongoing motion of the joint

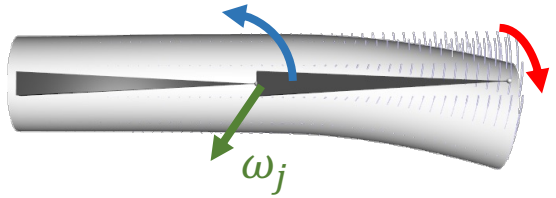
Linear velocity:
Translation *opposite to joint velocity*

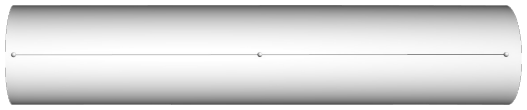
$$\varphi^{Lin} = -K^{Lin}\, L_j$$



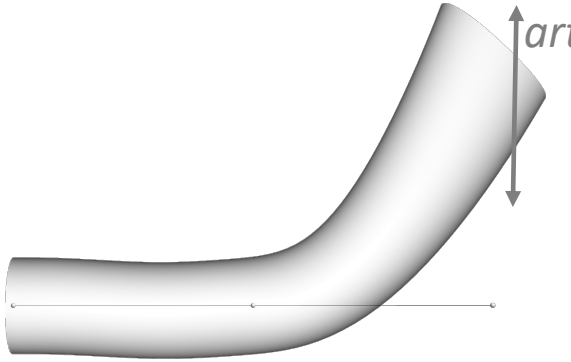
Angular velocity:
Rotation *opposite to joint angular velocity* Ω_j

$$\varphi^{Angl} = Rot\left(K^{Ang}||\Omega_j \times r_{ij}||, \Omega_j\right)$$

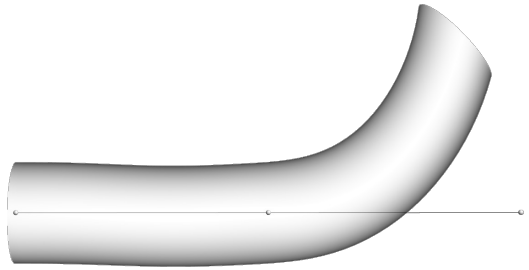




Initial cylinder

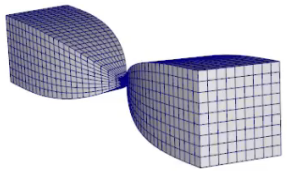


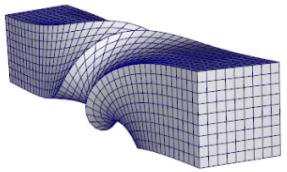
Using per-vertex velocity



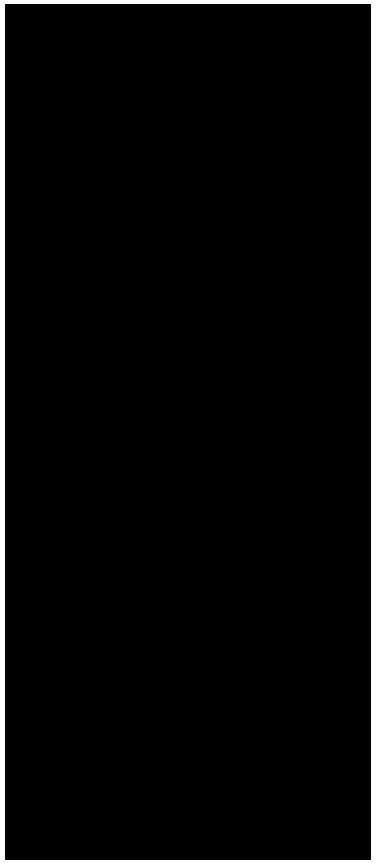
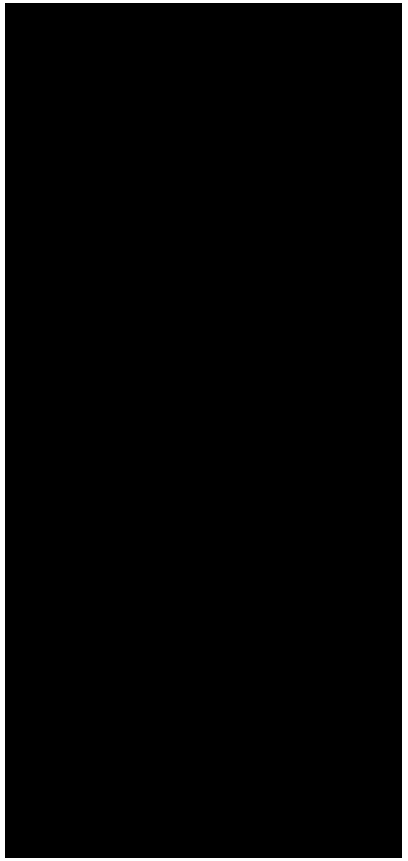
Our result:
Using joint velocity

DQS

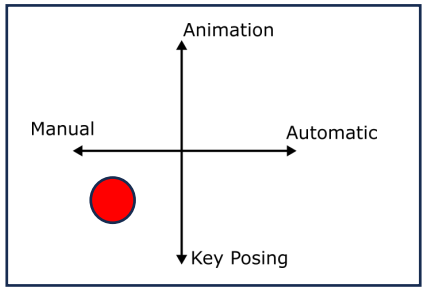




Betas

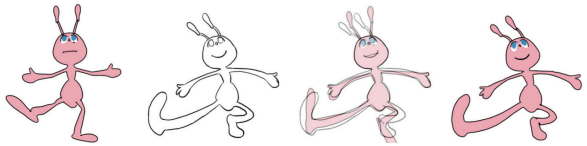
α_{ij}  β_{ij} 

Related works: Sketch based Approaches

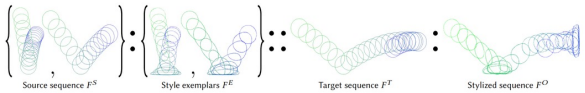


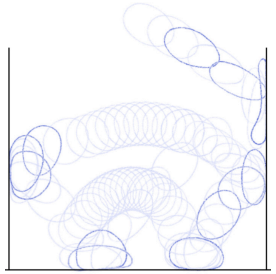
2D Shapes

[Li et al., Stylizing Motion with Drawings, SCA03]



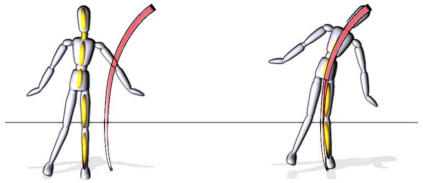
[Dvoroznak et al., Example-Based Expressive Animation of 2D Rigid Bodies, SIGGRAPH17]



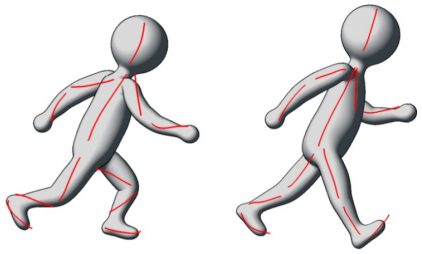


3D Poses

[Guay et al., The Line of Action, SIGGRAPH Asia13]

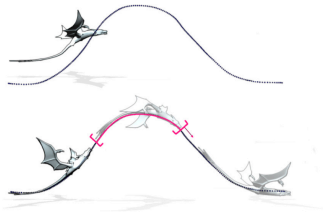


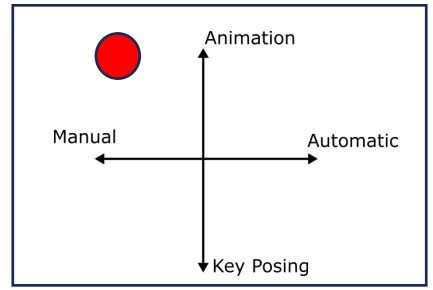
[Hahn et al., Sketch Abstractions for Character Posing. SCA15]



Space-time trajectories

[Guay et al., Space-time sketching of character animation, SIGGRAPH15]





[Choi et al., SketchiMo: Sketch-Based Motion Editing for Articulated Characters, SIGGRAPH16]

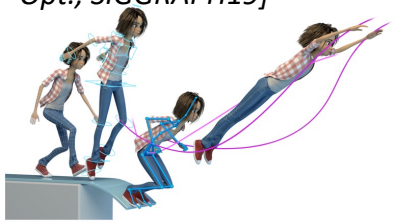


Extension to: Cycles, Key-Frames, VR

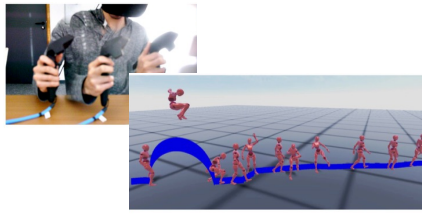
[Ciccone et al., Authoring Motion Cycles, SCA17]



[Ciccone et al., Tangent-Space Opt., SIGGRAPH19]



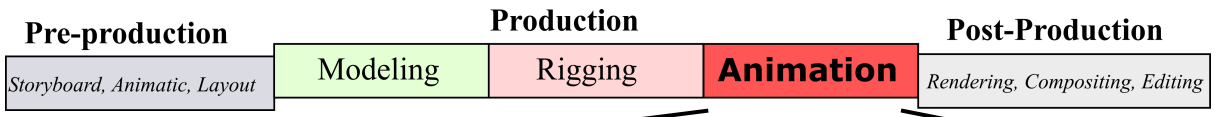
[Garcia et al., Spatial Motion Doodles, MIG 2019]



Current animation production methods

“Standard Production Pipeline”

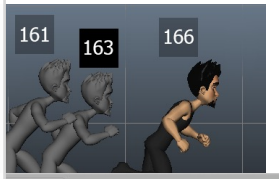
- ***Key Posing*** + ***Skinning***
- Then add Cartoon effects
  - (+) Highly controllable
  - (-) Teadious/time consuming

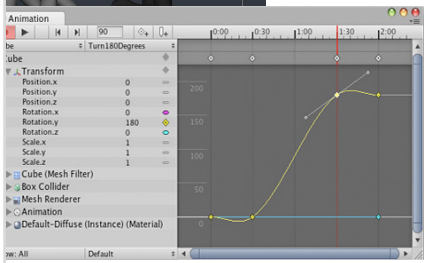


Can we develop Automatic 3D cartoon effect compatible with Production Constraints ?



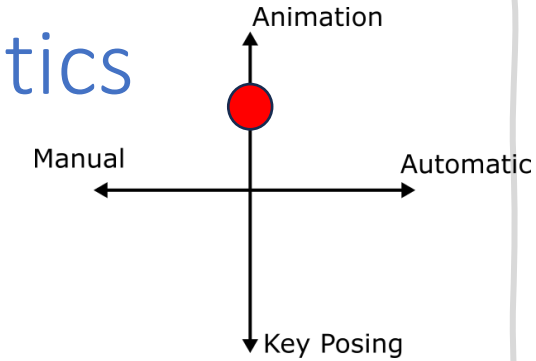
Automatic deformation & adaptation to skeletal hierarchy => rely on existing rig
Controlable
Real-time





“Animation” = key-framed with animation curves

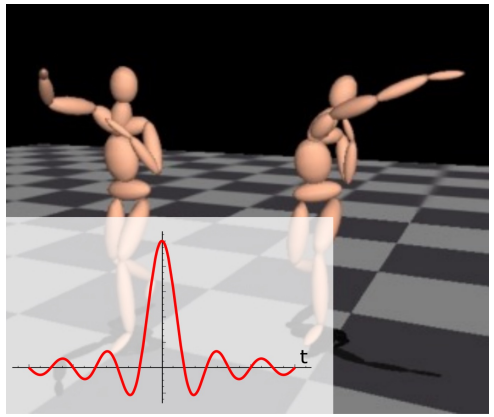
Related works: Intermediate – Expressive Kinematics



Automatic deformation, leverage Kinematics (velocity) Inputs

Time filter

[Wang et al., The cartoon animation filter. SIGGRAPH 06]
[Kass, Anderson. Animating Oscillatory Motion with Overlap: Wiggly Splines. SIGGRAPH 08]



+ Time control
- Spatial control

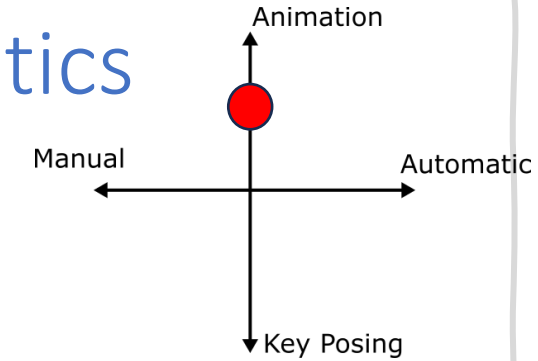
Parametric Bending

[Noble & Tang, Automatic Expressive Deformations for Stylizing Motion*, CGF06]*



+ Bending effect
- No Hierarchy

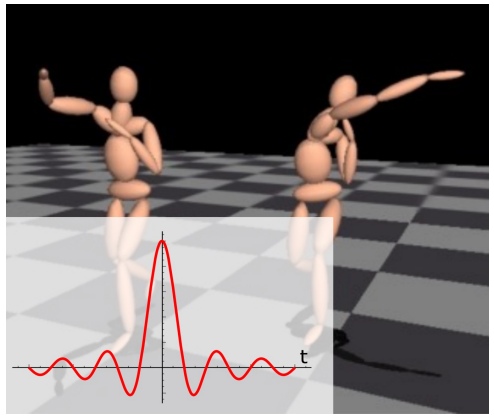
Related works: Intermediate – Expressive Kinematics



Automatic deformation, leverage Kinematics (velocity) Inputs

Time filter

[Wang et al., The cartoon animation filter. SIGGRAPH 06]
[Kass, Anderson. Animating Oscillatory Motion with Overlap: Wiggly Splines. SIGGRAPH 08]



+ Time control
- Spatial control

Dynamic Skinning: **Similar Positioning**

(+) Automatic Hierarchy
(+) Spatial (and time) control

Parametric Bending

[Noble & Tang, Automatic Expressive Deformations for Stylizing Motion*, CGF06]*



+ Bending effect
- No Hierarchy

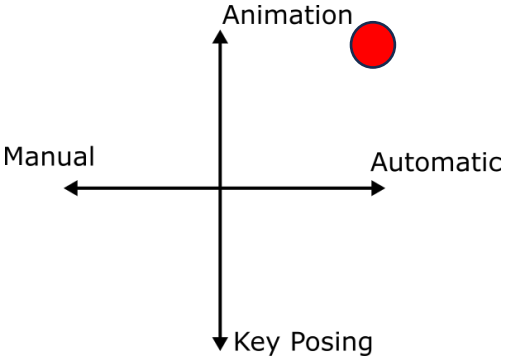
Related works: Physics-Based Simulation

Physics-based simulation

(+) provide automatically dynamics/inertia for skin deformation







Main challenges:

- *How to add/control expressive part*
- *Computational efficiency*

- Parameter fitting
  (limited possibility)

- Constrained optimization
  $\alpha E_{physics} + \beta E_{constraints}$

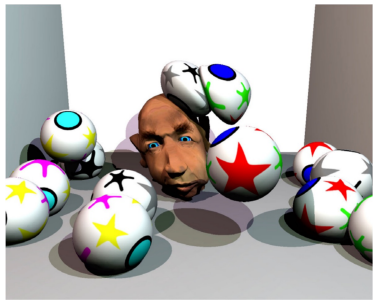
Reduced domains

[Xu16]
[Zhang20]
[Benchekroun23]
…



Positions Based

[Rumman14]
[Komaritzan18]
[Wu23]



Hybrid Models

[Larboulette05]
[Ji-Yong12]
[Rousselet18]
[Jacobson11]
…

