Rigging

Begriffsdefinition

- rig … Auftakelung, Manipulieren,
  [tech] Rüstung, Förderturm, Takelung
- Computer Animation
  - rig … skeleton for 3D model
  - rigging … build and fit skeleton into model
  - skinning … bind 3D model to a skeleton via bone weights

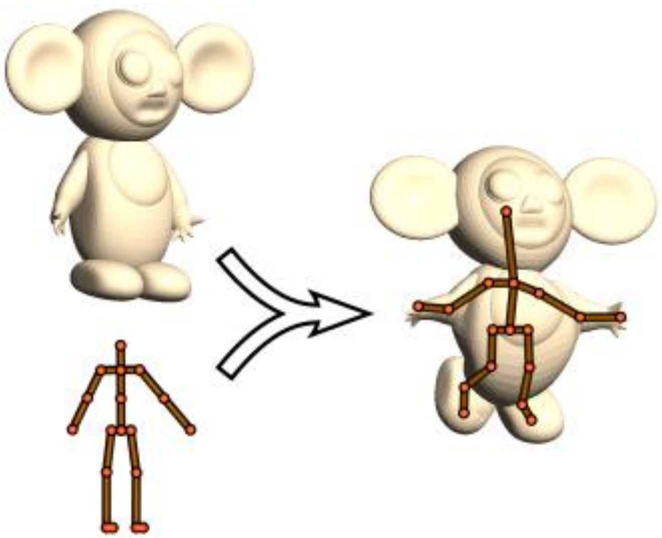
Overview

- Automatic Rigging – The Pinocchio System
- RigMesh – Sketch Based Modelling of rigged meshes
- Rigging from Animations

AUTOMATIC RIGGING

The Pinocchio System

- Paper: http://people.csail.mit.edu/ibaran/papers/2007-SIGGRAPH-Pinocchio.pdf
- Goal: fully automatic rigging and skinning of given mesh such that motion data can be mapped to mesh
- Input:
  - 3D mesh (close to standard pose),
  - skeleton (fixed topology), and
  - motion data (optional)
- Output:
  - skeleton embedding
  - vertex weights
- Software available:
  - https://github.com/elrond79/Pinocchio
  - Windows binary plus source code
  - example meshes, skeletons and motion data
  - allows rigging, skeleton export and animation



Overview of Pinoccio

- Discretization on graph built over medial sphere cover
- Discrete Embedding through coarse to fine (2 stage) discrete optimization of penalty function
- Embedding Refinement by continuous optimization of reduced penalty function
- Learn weights of penalty function by max margin method on good and bad example riggings
- Skinning weights computation through per bone solution of heat equation

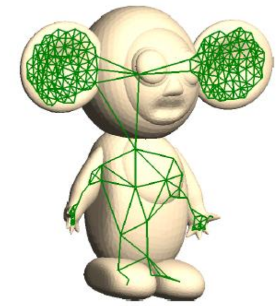
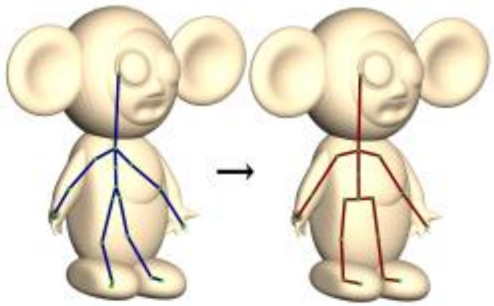
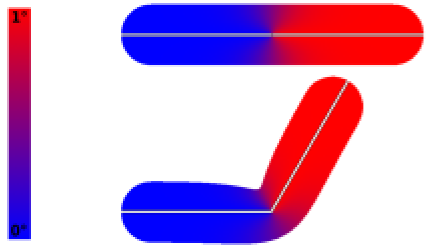


Figure 4: Constructed Graph



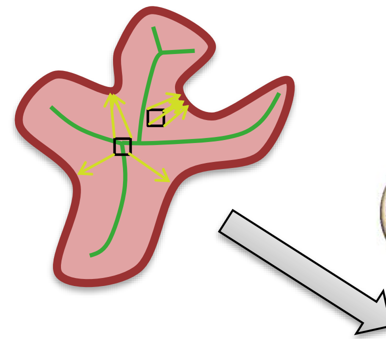
embedding refinement



Heat based weight computation

Discretization on Graph

- compute adaptive distance field (aDF) on octree
- approximatively sample medial axis from aDF discontinuities
  - examine aDF gradients at corners of voxels from finest octree level
  - construct medial axis sample if maximum angle between aDF gradients is greater than 120°
- construct sphere cover similar to poisson disk sampling on medial axis but take samples in order of decreasing sphere radius
- build neighbor graph on sphere centers with edges between overlapping spheres.



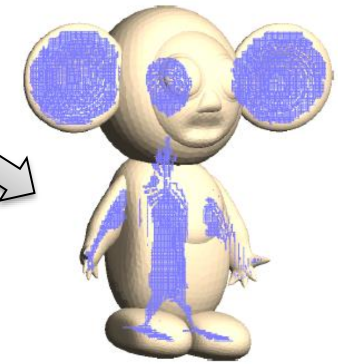


Figure 2: Approximate Medial Surface

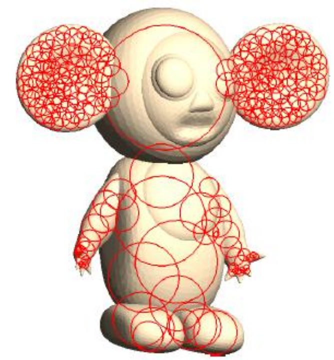


Figure 3: Packed Spheres

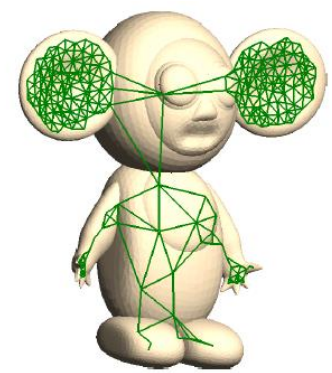
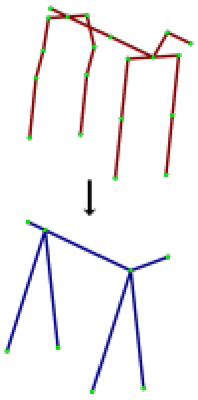
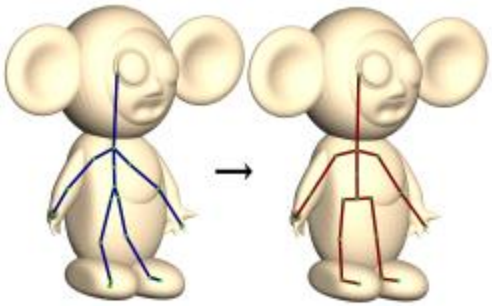


Figure 4: Constructed Graph

Discrete Embedding



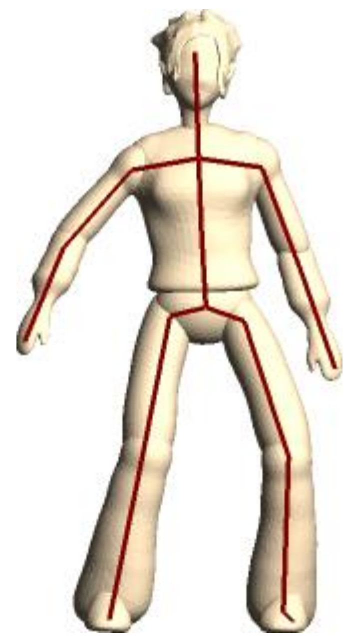
- reduce skeleton to one bone per chain
- define penalty function with nine weighted terms (see next slide)
- optimize reduced skeleton by A* algorithm
  - build priority queue of partial embeddings sorted by lower bound estimate of penalty function of extension to full embedding
  - extract best partial embedding, try all extensions and sort them into queue
  - discard extensions with very high lower bound penalty estimate
  - first full embedding found is optimum
- extend reduced skeleton and perform local continuous optimization (i.e. gradient descent) on simplified penalty function (in two more slides)



embedding refinement

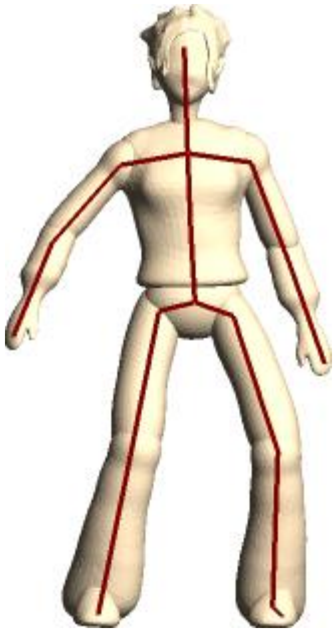
Terms of Discrete Penalty

1. short reduced bones (for each reduced bone compare
0.27 path length in reduced embedding to path length in
unreduced skeleton [estimate without optimization])
0.23 2. wrong direction reduced (compute angle between
bones in reduced embedding and input skeleton)
0.07 3. length unsymmetry (compute difference in length of
bones marked symmetric)
0.46 4. vertex sharing (compute number of vertices shared by
two kinetic chains)
0.14 5. feet (difference of y-value between foot y-value and
minimum joint y-value)
0.12 6. zero length (counts bones of zero length)
7. wrong direction unreduced (compute angle between
0.72 bones in unreduced embedding and skeleton)
8. extremity (penalize if there is a more extreme position
0.05 for chain end)
9. short Euclidean distance (compute difference between
0.33 path distances in embedding and Euclidean distances)



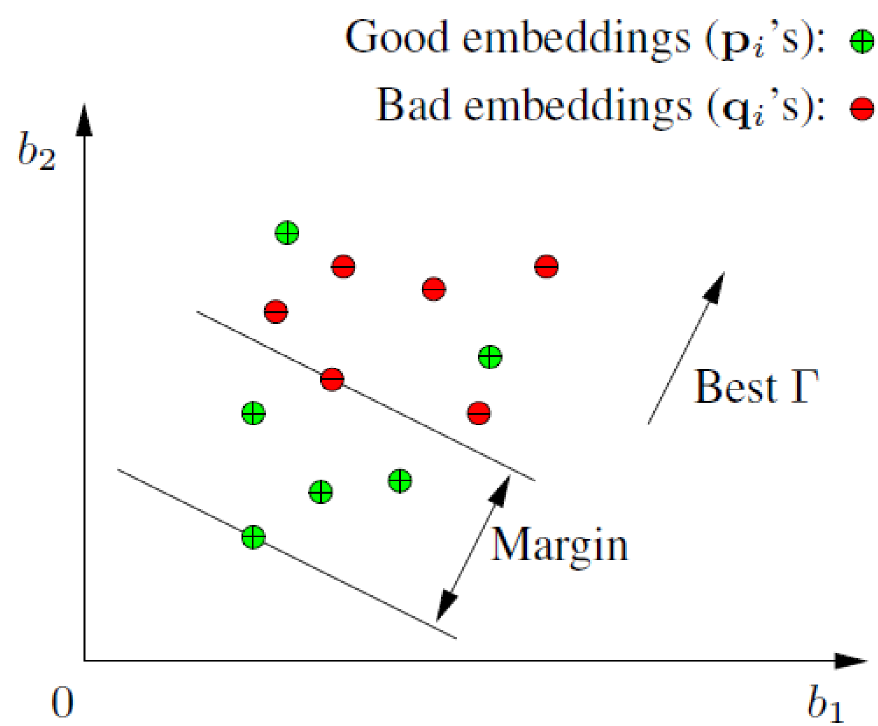
Terms of Refinement Penalty

1. center distance (sample bones and compute distance from sample to medial axis)
2. short bones (compare lengths in embedding and in skeleton)
3. wrong directions (compare bone directions in embedding and skeleton)
4. unsymmetry (compare length of symmetric bones)



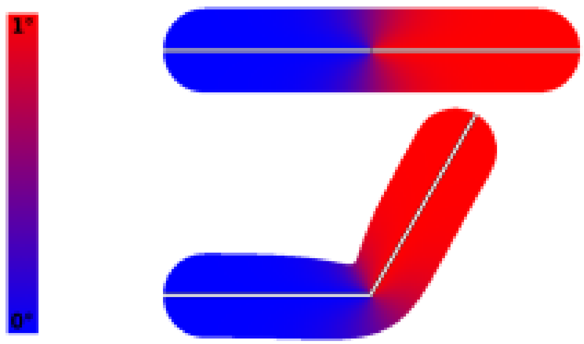
Max Margin Learning

- generate a large number of positive and negative examples (lots of work!)
- penalty terms define points in 9D / 4D weighting space
- find weight vector Γ that defines coordinate direction such that difference along this direction between minimal coordinate of bad examples and *minimum* coordinate of good examples is maximized
- leads to non convex optimization problem that is solved by randomly sample space and downhill simplex method on each sample



Heat Based Skinning

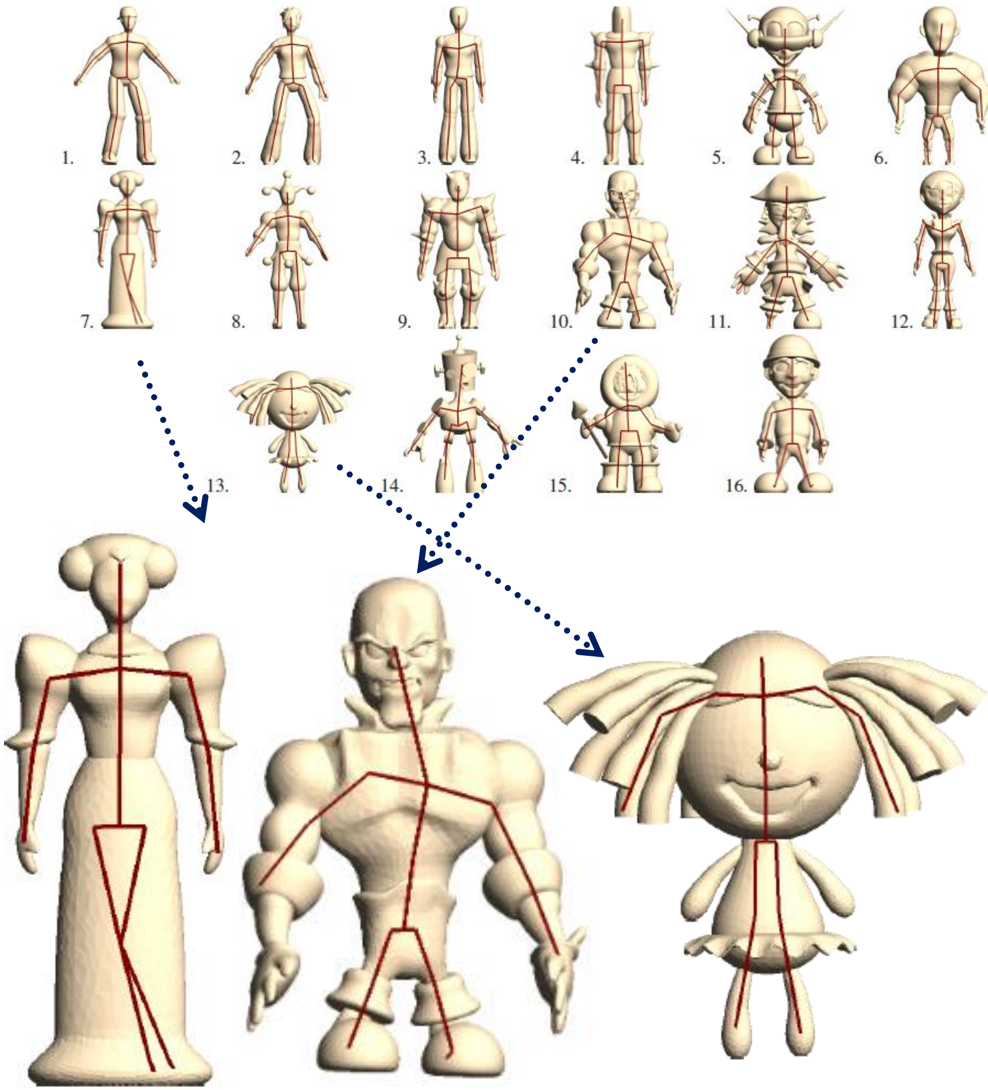
- uses linear blend skinning
- define vertex weights for each bone i by setting heat at bone i to 1 and 0 on all other bones
- Solving heat equation on shape gives vertex weights
- approximate solution by simulating heat equation on surface
  - set initial heat values of some vertices
  - use surface Laplacian to discretize heat equation with internal sources
  - leads to sparse linear system that can be solved i.e. with TAUCS library



Heat based weight computation

Results and Limitations

- typical runtime of 30s
- successful on 13 of 16 test models
- 1-click correction sufficient in all failed cases
- weights generalize to quadped skeletons
- software plugins available (Blender: "Bone Heat" and Maya: PM_heatWeight)
- limitations
  - skeleton must be given
  - very thin limbs can cause problems
  - degree 2 joints such as knees can be hard to find



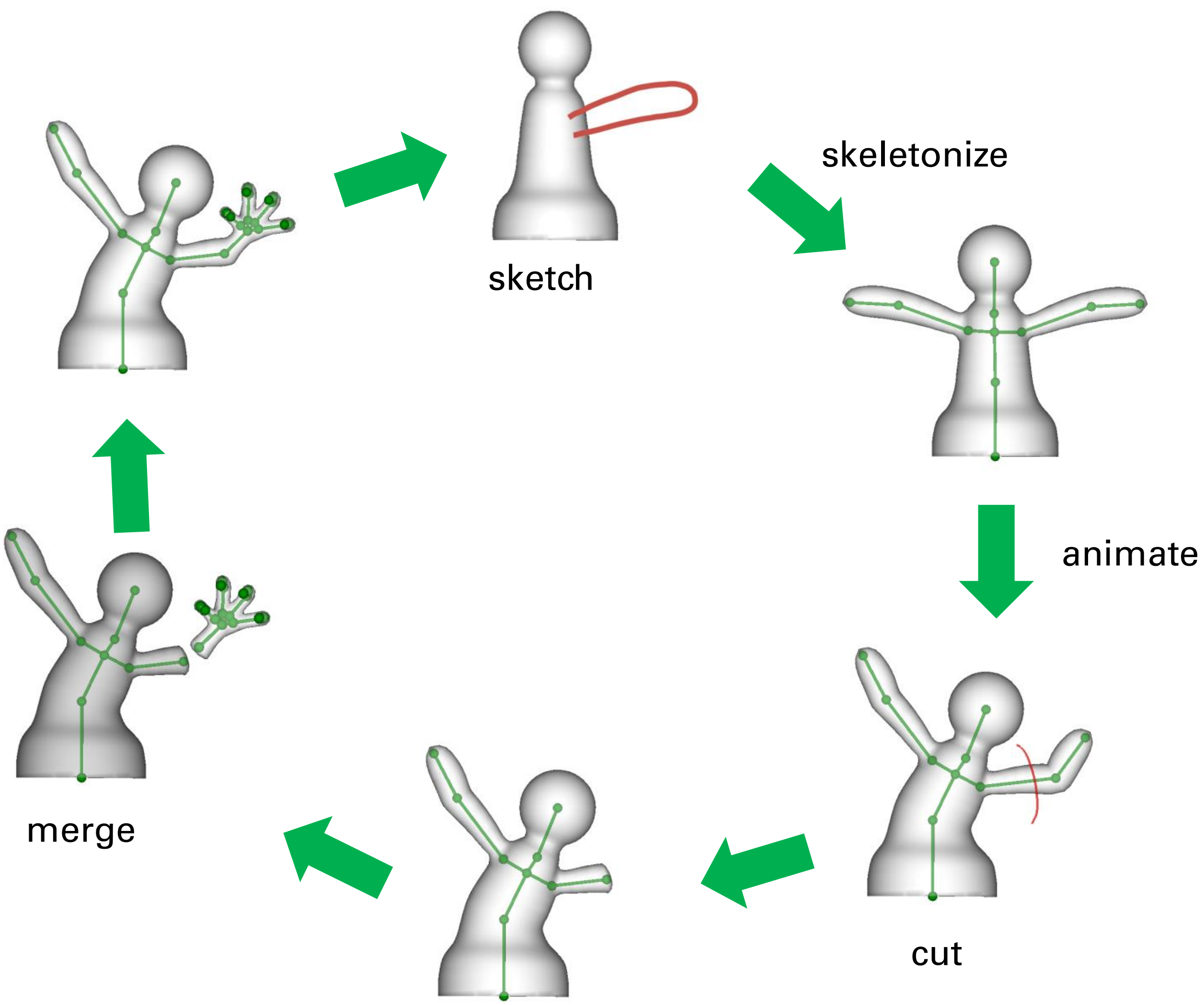
failed examples

RIGMESH

Overview of RigMesh

- **Problem:** rigging is post processing and needs to be redone if further edit operations are necessary
- **Idea:** incorporate rigging into sketch based modeling
- sketch based interface:
  - draw contours
  - cut, copy and paste parts
  - edit skeleton
  - skeleton based animation
- advantage:
  - no post processing necessary
  - Skeleton based animation possible
  - animation quality can be tested early in modeling process
- implementation:
  - makes Pinocchio system incremental
  - implements skeletonization based on generalization of Douglas Peucker algorithm

Example



RigMesh:

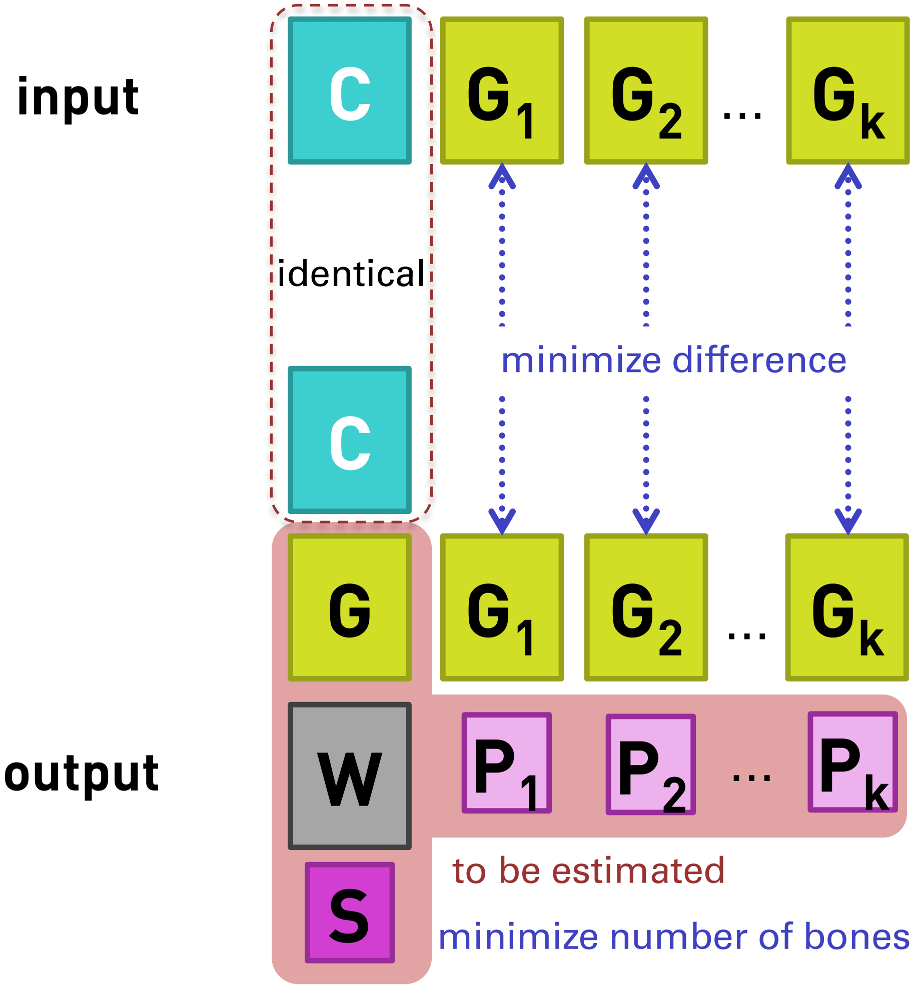
Automatic Rigging for Part-Based Shape Modeling and Deformation

cs.gmu.edu/~ygingold/rigmesh

www.youtube.com/watch?v=1prInV9ZNY0

RIGGING FROM ANIMATIONS

Rigging from Animations



- G_k Geometry
- C Connectivity
- W Vertex Weights
- P_k Skeleton Pose
- S Skeleton

Rigging From Animations

- Input: animated mesh
- Output: bones, vertex weights, bone motion
- Advantages:
  - compression
  - suitable for real-time animation of herds
- Discussed Approaches
  - Skinning mesh animations: http://graphics.cs.cmu.edu/projects/sma/
  - Fast and Efficient Skinning of Animated Meshes http://www.jarmilakavanova.cz/ladislav/papers/sam-eg10/sam-eg10.htm

Matrix Decomposition

- Singular Value Decomposition (SVD):
  - any $n \times m$-matrix $\boldsymbol{C}$ can be decomposed into an orthonormal $n \times n$-matrix $\boldsymbol{U}$, a diagonal $n \times m$-matrix $\boldsymbol{D}$ and an orthonormal $m \times m$-matrix $\boldsymbol{V}$, such that:

$$\boldsymbol{C} = \boldsymbol{U} \cdot \boldsymbol{D} \cdot \boldsymbol{V}^T$$

  - if $\boldsymbol{C}$ is quadratic and symmetric, we have $\boldsymbol{U} = \boldsymbol{V}$ and the decomposition is called Eigenvalue decomposition $\boldsymbol{C} = \boldsymbol{U} \cdot \boldsymbol{D} \cdot \boldsymbol{U}^T$
  - In case of multiple Eigenvalues decomposition is not unique
- Polar Decomposition:
  - Any quadratic $n \times n$-matrix $\boldsymbol{C}$ can be uniquely decomposed into a positive semi-definite $n \times n$-matrix $\boldsymbol{A}$ and an orthonormal $n \times n$-matrix $\boldsymbol{R}$, such that:

$$\boldsymbol{C} = \boldsymbol{R}\boldsymbol{A} = \boldsymbol{U} \cdot \boldsymbol{D} \cdot \boldsymbol{V}^T = \Big(\underbrace{\boldsymbol{U}\boldsymbol{V}^T}_{\boldsymbol{R}}\Big)\Big(\underbrace{\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T}_{\boldsymbol{A}}\Big)$$

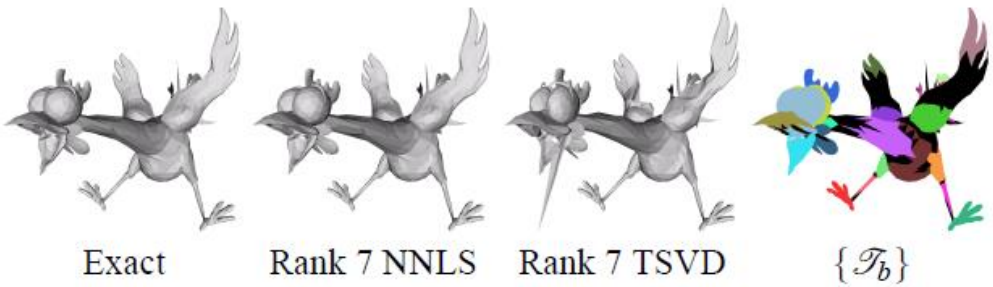
  - here $\boldsymbol{R}$ is rotation, that approximates $\boldsymbol{C}$ best with respect to Frobenious norm
  - Iterative approximation: $\boldsymbol{R}_0 = \boldsymbol{C}, \quad \boldsymbol{R}_{i+1} = \frac{1}{2}\left(\boldsymbol{R}_i + \left(\boldsymbol{R}_i^T\right)^{-1}\right)$

Skinning Mesh Animations (SMA)

Overview

- segment triangles into rigid and flexible parts by examining their time evolutions
- compute bone transformations as rigid transform (or affine transform for flexible bones)
- estimate vertex weights by truncated or non-linear least squares fitting
- use SVD on errors to add progressive corrections

rigid triangles in different colors



SMA – triangle segmentation I

- for each triangle compute feature vector consisting of concatenation of per frame rotations between triangle in initial pose and frame pose
- transformation is in general an affine transformation that is split with polar decomposition into rotation and symmetric matrix
- Perform mean shift clustering in rotation sequences

$$h = 9k\varepsilon \text{ and } \varepsilon = 0.05$$

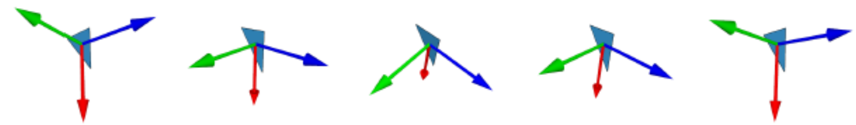


Figure 3: **Triangle rotation sequences** *as estimated using the polar decomposition. Rotation sequences represent each triangle motion as a high-dimensional point for subsequent mean shift clustering to estimate near-rigid components.*

$$\mathbf{z}_j = \left(\mathrm{vec}(R_j^1),\ \ldots,\ \mathrm{vec}(R_j^S)\right) \quad (3)$$

where $\mathrm{vec}(R) : \mathbb{R}^{3x3} \rightarrow \mathbb{R}^9$ converts the row-ordered 3×3 rotation matrix, R, to a row-major 9-vector.

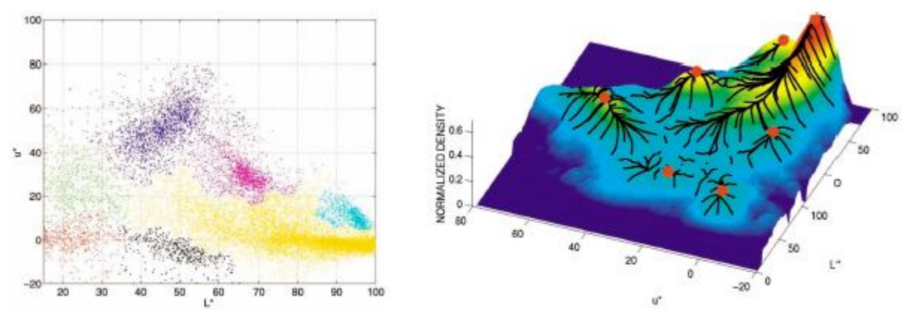


Figure 4: **Mean shift clustering of 2D points** *(images courtesy of [Comaniciu and Meer 2002] ©IEEE 2002) (Left) Input 2D points, with color-coded cluster output; note the interesting oblong cluster shapes. (Right) Related density field, with trajectories from the mean shift gradient ascent algorithm. Red dots indicate the final mode centers used for proximity-based classification of mean-shifted points.*

SMA – triangle segmentation II

rigid bone estimation

- compute for each bone in each frame a matrix by averaging over all rotation matrices of bone triangles
- compute rotation with polar decomposition
- compute translation with area-weighted least squares fit to triangle centers

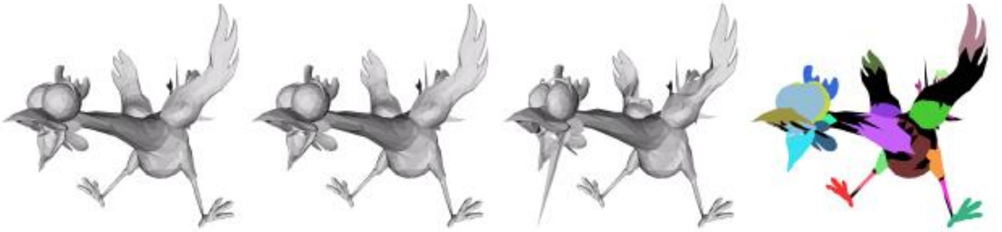
flexible bone estimation

- directly do least squares fit of affine transformation to triangle centers
- this is used in paper

Estimate Vertex Weights

- firstly, bone influences are estimated
  - user specifies maximum number b of non zero bone weights per vertex
  - greedily choose per vertex the b bones with the smallest approximation error over the animation

- estimate vertex weights
  - for each vertex the linear blend skin positions should represent the animation as well as possible
  - this leads per vertex to a linear least squares problem with constraint that per vertex weights sum to 1
- authors empirically found that solving LLS problems unconstrained with post-normalization is sufficient
- Avoiding overfitting
  - The fit can yield large negative weights leading to overfitting (spiking artefacts in new bone poses)
  - This can be avoided by truncated least squares or even better by non negative least squares (NNLS)



Exact Rank 7 NNLS Rank 7 TSVD $\{\mathcal{T}_b\}$

Skinning mesh animations

graphics.cs.cmu.edu/projects/sma

Fast and Efficient Skinning of Animated Meshes

L. Kavan, P.-P. Sloan, C. O'Sullivan

Disney Interactive Studios
Trinity College Dublin

www.youtube.com/v/e0rugcfR8K4

F&ESAM – Matrix Formulation

- Kavan et al. [2010] use linear blend skinning to decompose the matrix $\boldsymbol{A}$ containing the k time steps of the m vertex positions into a bone transformation matrix $\boldsymbol{T}$ for n bones and a matrix $\boldsymbol{X}$ containing the initial pose and the vertex weights:

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

homogeneous representation of blend skinning with normalized weights

$$3k\ \underbrace{\overset{\longleftarrow\ m\ \longrightarrow}{\begin{pmatrix} \boldsymbol{p}_1^1 & \boldsymbol{p}_2^1 & \cdots & \boldsymbol{p}_m^1 \\ \boldsymbol{p}_1^2 & \boldsymbol{p}_2^2 & \cdots & \boldsymbol{p}_m^2 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{p}_1^k & \boldsymbol{p}_2^k & \cdots & \boldsymbol{p}_m^k \end{pmatrix}}}_{\boldsymbol{A}} = \underbrace{\overset{\longleftarrow\ 4n\ \longrightarrow}{\begin{pmatrix} \boldsymbol{M}_1^1 & \boldsymbol{M}_2^1 & \cdots & \boldsymbol{M}_n^1 \\ \boldsymbol{M}_1^2 & \boldsymbol{M}_2^2 & \cdots & \boldsymbol{M}_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{M}_1^k & \boldsymbol{M}_2^k & \cdots & \boldsymbol{M}_n^k \end{pmatrix}}}_{\boldsymbol{T}} \underbrace{\overset{\longleftarrow\ m\ \longrightarrow}{\begin{pmatrix} w_{11}\tilde{\boldsymbol{p}}_1^0 & w_{12}\tilde{\boldsymbol{p}}_2^0 & \cdots & w_{1m}\tilde{\boldsymbol{p}}_m^0 \\ w_{21}\tilde{\boldsymbol{p}}_1^0 & w_{22}\tilde{\boldsymbol{p}}_2^0 & \cdots & w_{2m}\tilde{\boldsymbol{p}}_m^0 \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1}\tilde{\boldsymbol{p}}_1^0 & w_{n2}\tilde{\boldsymbol{p}}_2^0 & \cdots & w_{nm}\tilde{\boldsymbol{p}}_m^0 \end{pmatrix}}}_{\boldsymbol{X}}$$

animation matrix
row triples are mesh frames
columns are vertex trajectories

bone transformation matrix

$$R^{3\times 4} \ni \boldsymbol{M}_i^t = \tilde{\boldsymbol{T}}_i^{0\to t}\Big|_{xyz}$$

weights and initial pose matrix

F&ESAM – Trajectory Space Reduction

- The skinning problem can be formulated as matrix decomposition problem:
  decompose $\boldsymbol{A}$ *into* $\boldsymbol{TX}$ *such that per vertex a limited number of bone weights are unequal zero*
- To improve efficiency the problem is solved in a reduced $d \ll m$ - dimensional trajectory space. For this the m columns of $\boldsymbol{A}$ are approximated by d orthogonal columns in a $3k$xd-dimensional matrix $\boldsymbol{B}$ and a dxm-dimensional matrix $\boldsymbol{C}$ with d trajectory coefficients per vertex. For a given approximation error ε $\boldsymbol{B}$, $\boldsymbol{C}$ and d are computed to minimize d that fulfills:
  $$||\boldsymbol{A} - \boldsymbol{BC}||_F < \varepsilon$$
- This can be solved by SVD, but Kavan et al. propose an alternate method that yields larger d but in much shorter time, such that the overall runtime is reduced significantly.

F&ESAM – Matrix Decomposition

Initialization:

- the first animation frame is used as initial pose $\underline{\boldsymbol{p}}_j^{r=0}$
- a simple region growing triangle clustering is used to set initial bone weights $w_{ij}^{r=0}$

$$\boldsymbol{A} = \boldsymbol{T} \cdot \underbrace{\begin{pmatrix} w_{11}\tilde{\boldsymbol{p}}_1^0 & w_{21}\tilde{\boldsymbol{p}}_2^0 & \cdots & w_{m1}\tilde{\boldsymbol{p}}_m^0 \\ w_{12}\tilde{\boldsymbol{p}}_1^0 & w_{22}\tilde{\boldsymbol{p}}_2^0 & \cdots & w_{m2}\tilde{\boldsymbol{p}}_m^0 \\ \vdots & \vdots & \ddots & \vdots \\ w_{1n}\tilde{\boldsymbol{p}}_1^0 & w_{2n}\tilde{\boldsymbol{p}}_2^0 & \cdots & w_{mn}\tilde{\boldsymbol{p}}_m^0 \end{pmatrix}}_{\boldsymbol{X}}$$

Alternating Optimization

1. $\boldsymbol{T}_r$ is computed from least squares problem
$$\boldsymbol{T}_r \boldsymbol{X}_r \boldsymbol{X}_r^T = \boldsymbol{A} \boldsymbol{X}_r^T$$
2. Given $\boldsymbol{T}_r$ and weights w_{ij} each rest pose location $\underline{\boldsymbol{p}}_j^0$ is computed from least squares problem (see paper)
3. Given $\boldsymbol{T}_r$ and rest pose locations $\underline{\boldsymbol{p}}_j^0$ four convex weights are optimized per vertex with specialized solver given in paper

- iterate 1.-3. till convergence

Figure 5: *Our clustering technique produces crude initial segmentation (left), but this is fixed in subsequent optimization (right) (samba dataset, see Table 2).*

REFERENCES

References

- Ilya Baran and Jovan Popović. 2007. Automatic rigging and animation of 3D characters. *ACM Trans. Graph.* 26, 3
- Péter Borosán, Ming Jin, Doug DeCarlo, Yotam Gingold, and Andrew Nealen. 2012. RigMesh: automatic rigging for part-based shape modeling and deformation. *ACM Trans. Graph.* 31, 6
- Skinning mesh animations: http://graphics.cs.cmu.edu/projects/sma/
- L. Kavan, P.-P. Sloan, C. O'Sullivan, Fast and Efficient Skinning of Animated Meshes, CGF 2010 http://www.jarmilakavanova.cz/ladislav/papers/sam-eg10/sam-eg10.htm