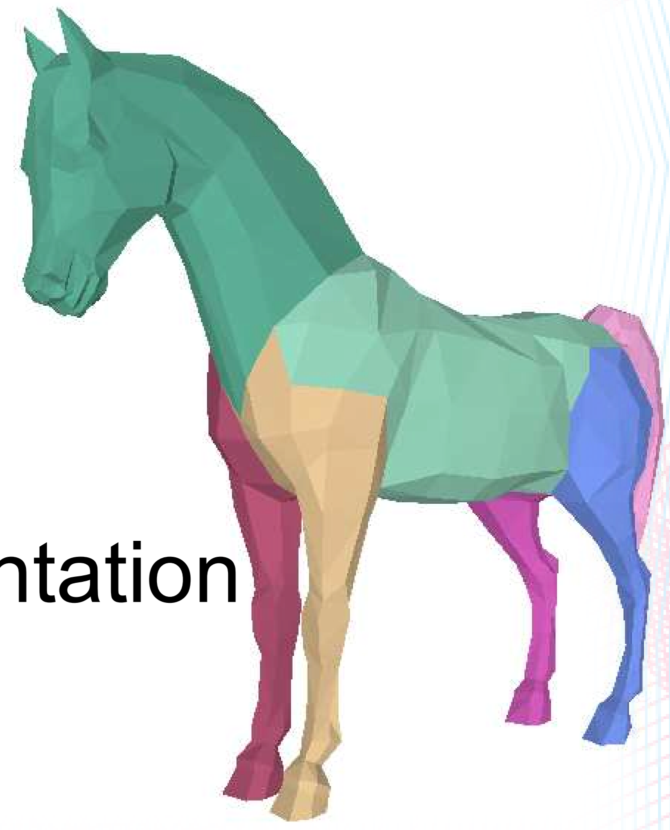
Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts Katz and Tal, Siggraph 2003

CS284 Paper Presentation

Arpad Kovacs

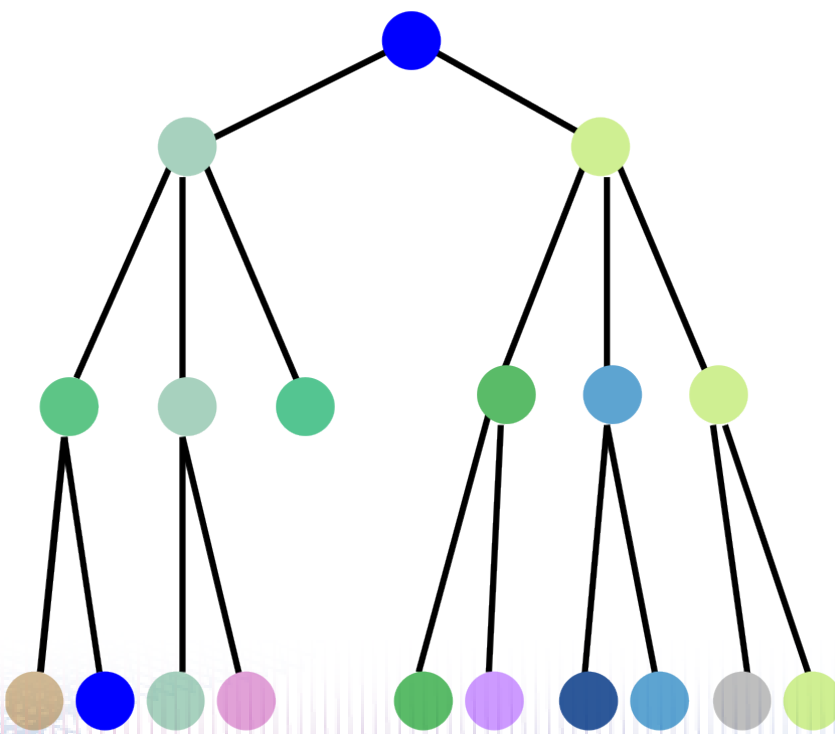
2009.11.16

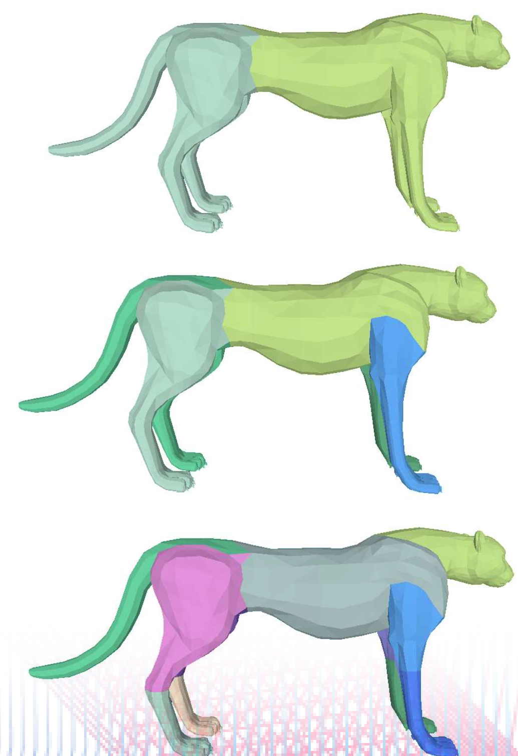


What is an Ideal Decomposition?

K-way decomposition = Segment a mesh into k connected, meaningful and **hierarchical** patches

Facewise-disjoint = Each face should belong to exactly 1 patch, with no overlapping faces.



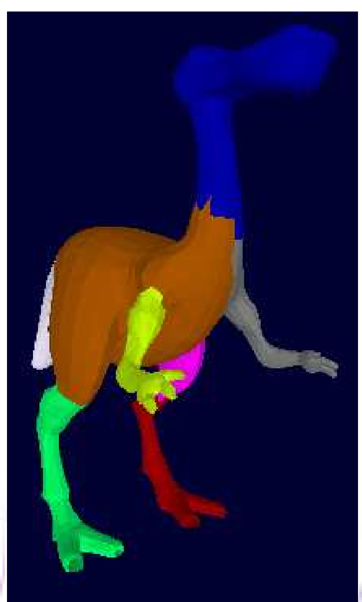
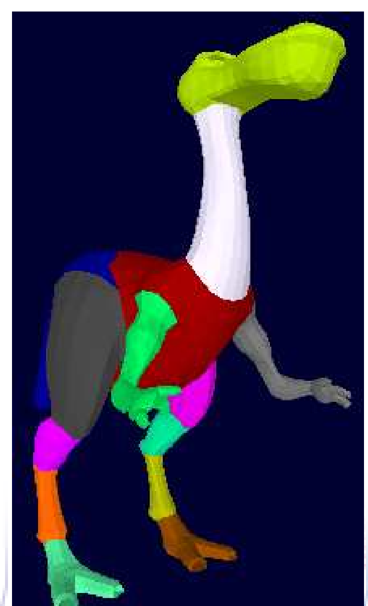


Advantages of This Algorithm

Handles general meshes (does not have to be 2-manifold or closed or triangulated)

Avoids over segmentation (too many patches that do not convey meaningful information)

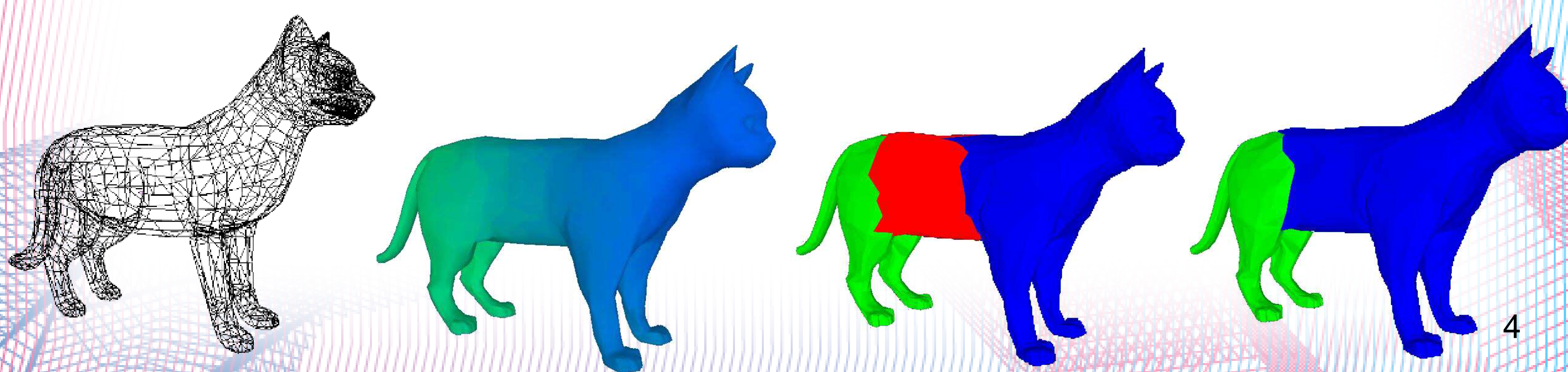
Smooth boundaries between patches, no jagged edges


Chazelle Katz

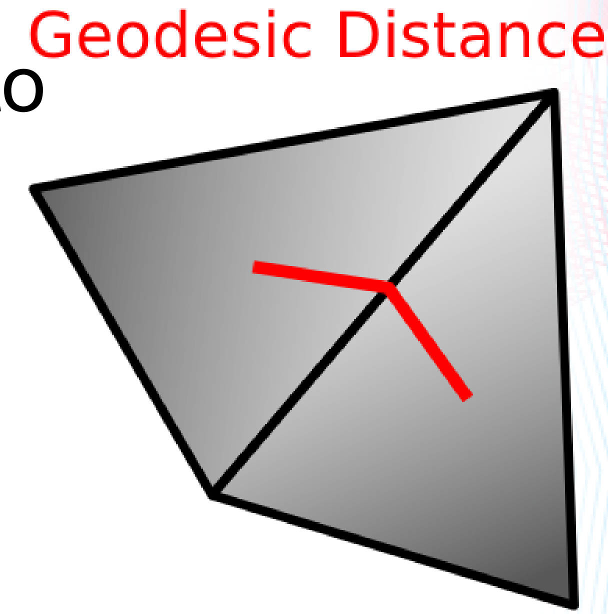
Algorithm Overview

1. Find distances between all pairs of faces in mesh
2. Calculate probability of face belonging to each patch
3. Refine probability values using iterative clustering
4. Construct exact boundaries between components



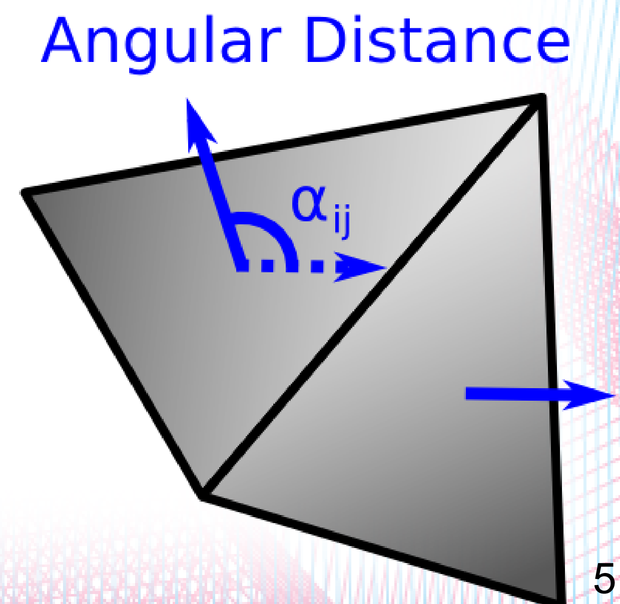
Distance Between Adjacent Faces

Rationale: Distant faces less likely to belong to same patch



Geodesic Distance: shortest path along the surface between two adjacent faces' centers of mass.

Angular Distance: consider angle between face normals and concavity
If concave, η=1, else η=small +value
    Cut at regions of deep concavity



$\text{AngularDistance}(\alpha_{ij}) = \eta(1-\cos(\alpha_{ij}))$

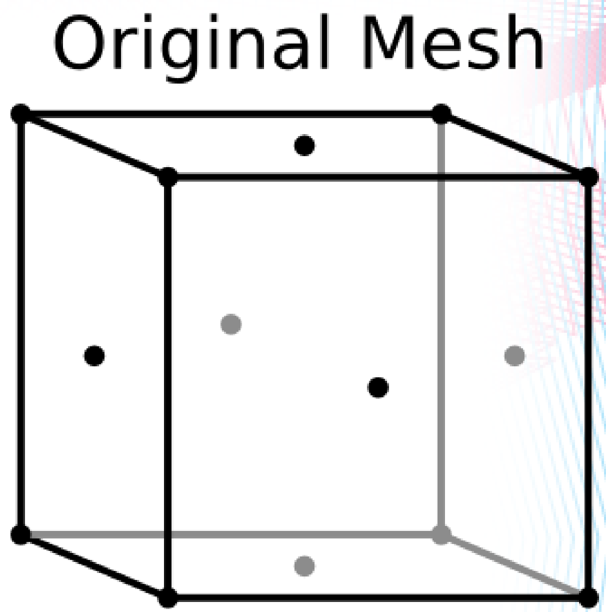
Dual Graph

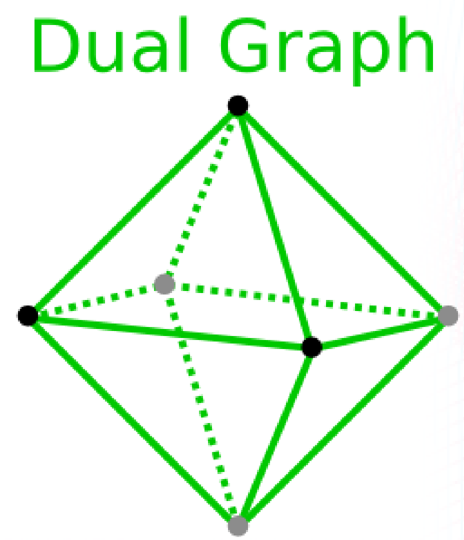
Each face in the mesh is a **vertex**
Arcs join <u>adjacent</u> faces' vertices

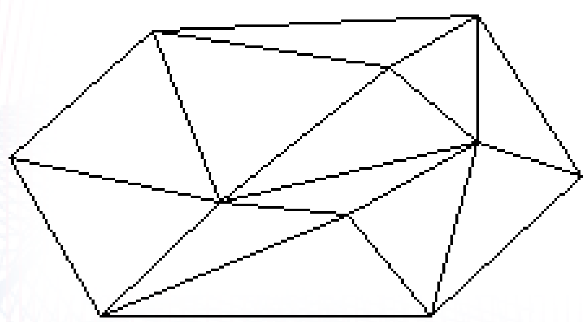
Arclength between vertices f_i, f_j :

$$\frac{\delta \cdot geodesicDistance(f_i, f_j)}{averageGeodesicDistance} + \frac{(1-\delta) \cdot angularDistance(f_i, f_j)}{averageAngularDistance}$$

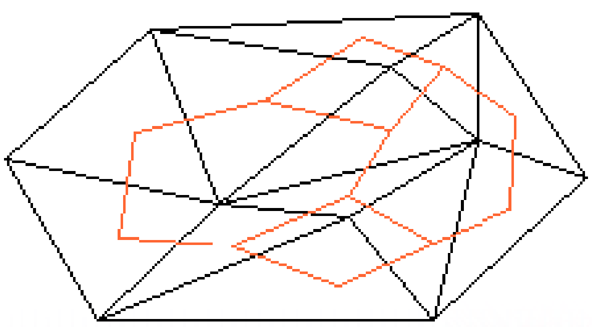
δ determines relative weight of geodesic vs angular distance







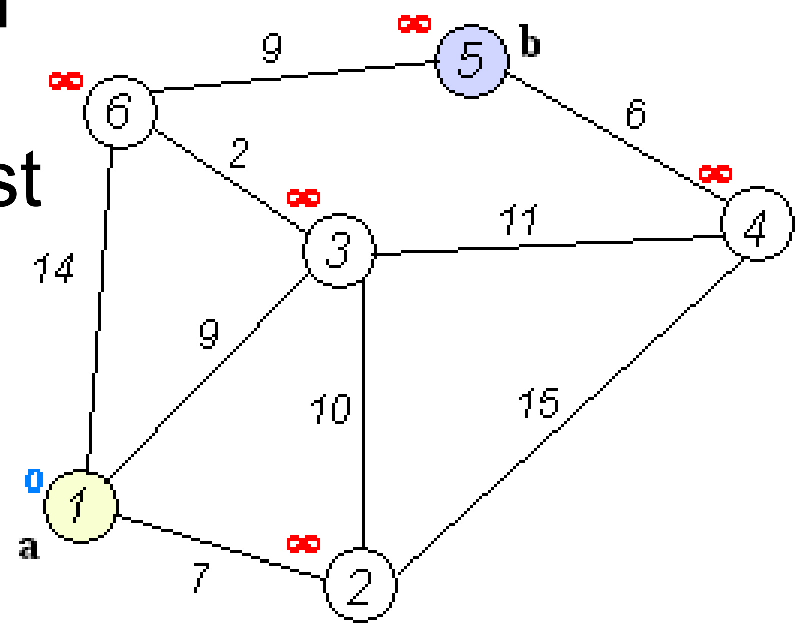
Simplex Mesh 2D



Dual Mesh

Dijkstra's Shortest Path Algorithm

1. Initialize distances: starting node = 0, others = ∞
2. Mark all nodes as unvisited.
   Set current = start node
3. Calculate cumulative distance to each of current node's unvisited neighbors from starting node
4. Mark current node = visited
   Update current node = unvisited node with smallest distance from start node.
5. If unvisited nodes remain,
   Goto step 3.
   Otherwise done



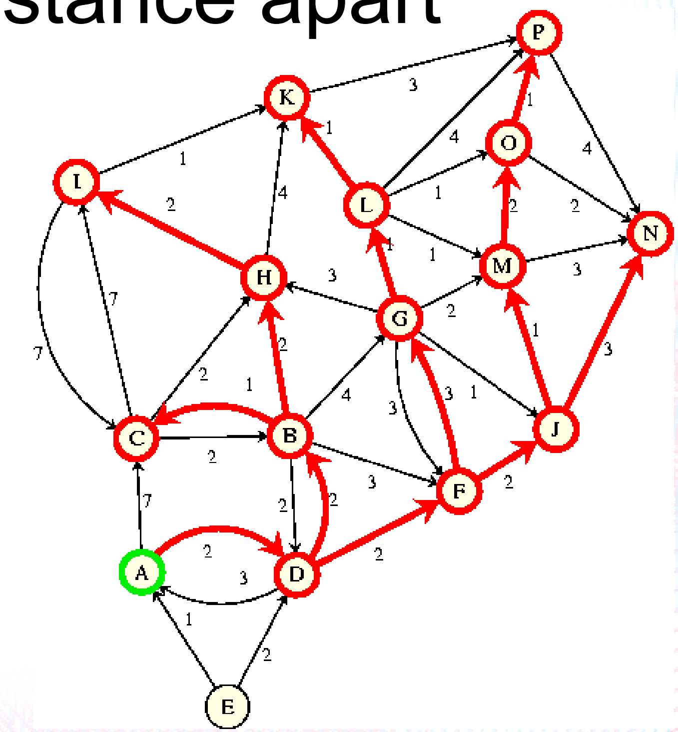
All-Pairs Shortest Path Algorithm

Calculate distances between each pair of points on graph, based on distances between adjacent points.

Small arc length = more connected; points on different components ∞ distance apart

Run Dijkstra's Algorithm from each node to every other node

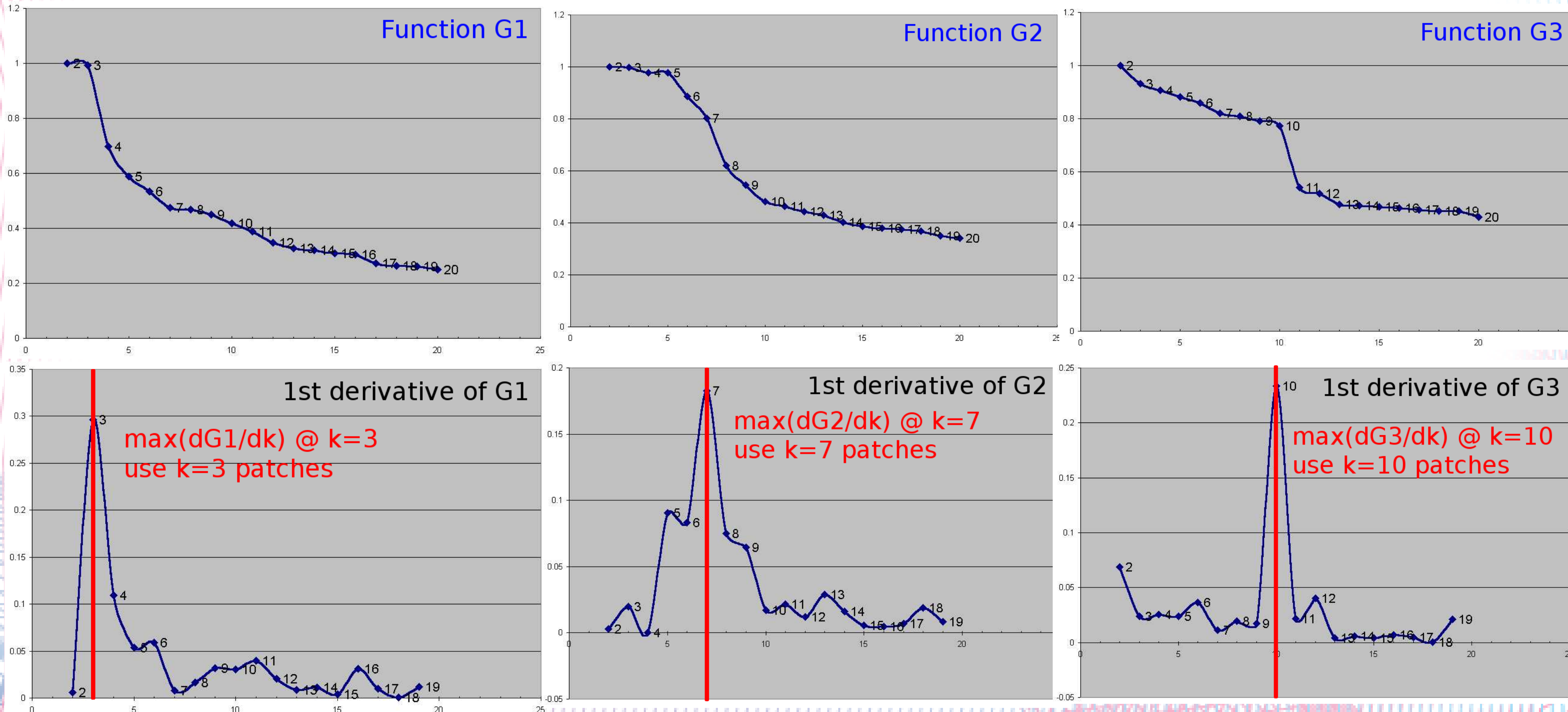
*(can also use Floyd-Warshall Algorithm for better efficiency)



How Many Patches/Seed Faces?

Maximize benefit from each additional cut/seed face

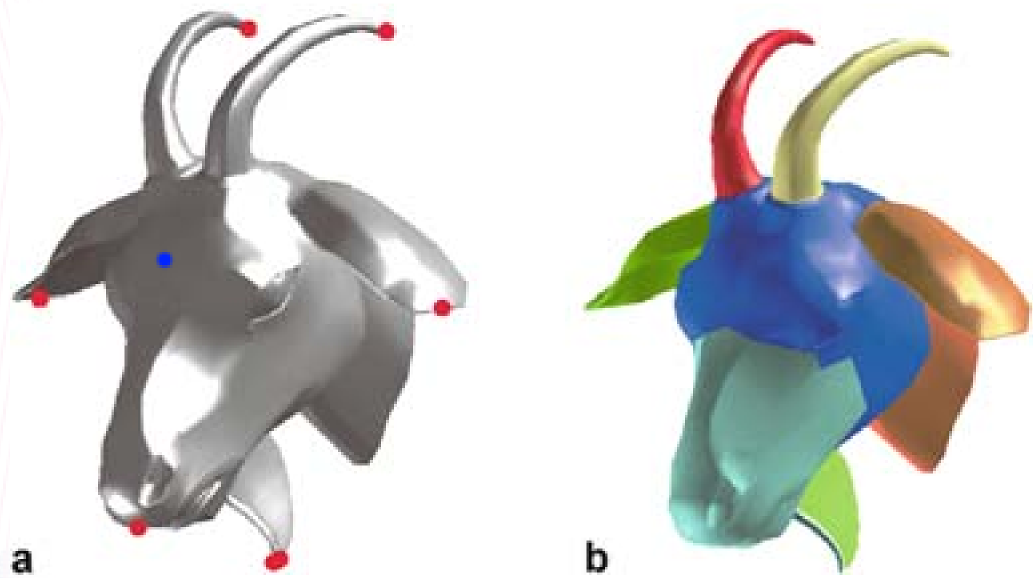
Maximize 1st derivative of G(k) describing how close new seed k is from already assigned seed

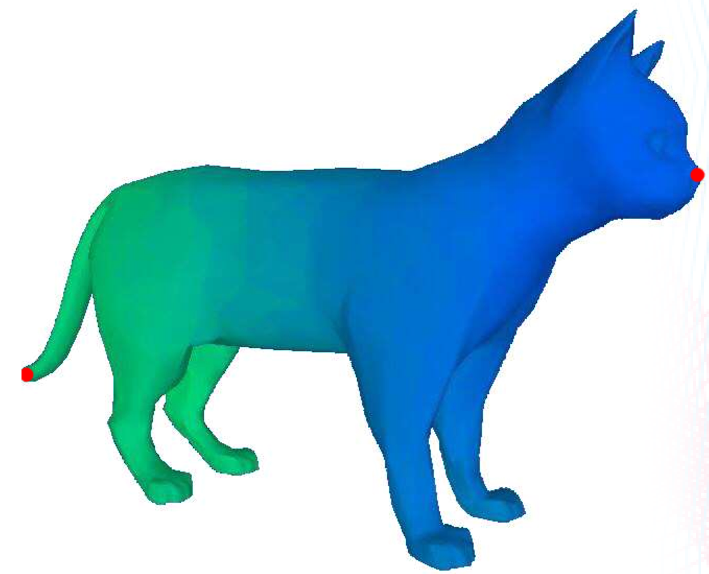


Selecting Seed Faces

1st seed closest to all other faces (represents body)*

Other faces maximize min-distance to already assigned seeds (stay far away from existing seeds)





* Special case for k=2, seed faces are ends of max(shortest path between vertex pairs)

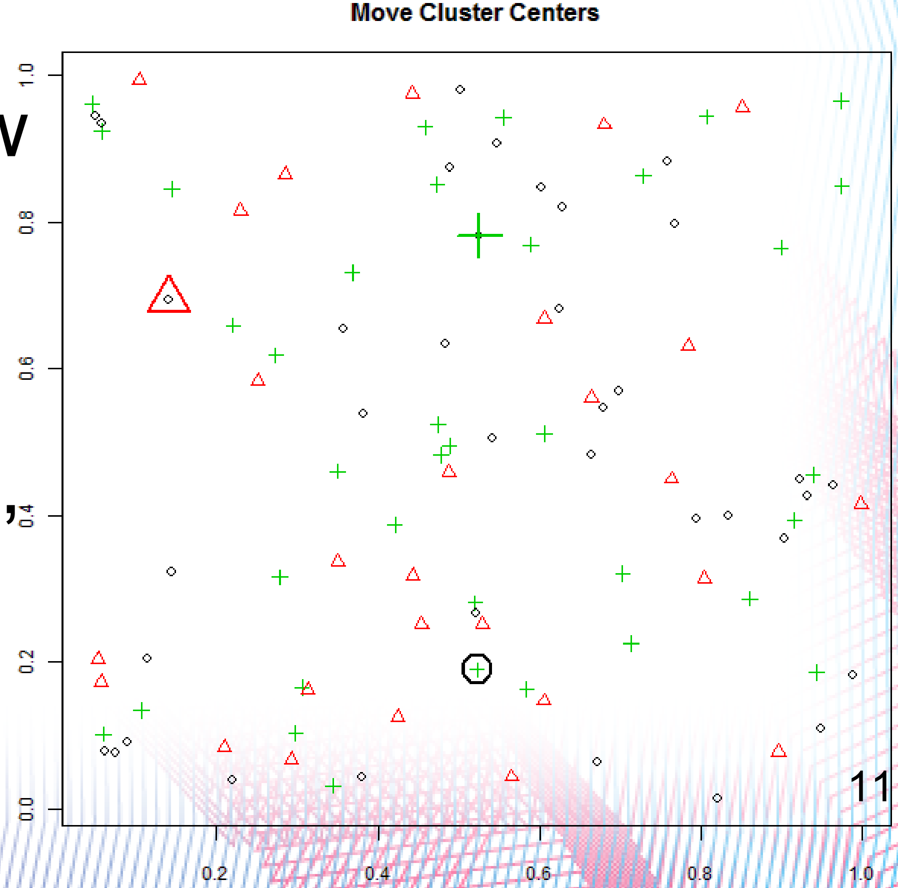
K-Means Clustering

Partition all vertices into k sets so that the within-cluster sum of distances squared is minimized.

1. **Assignment**: Assign each vertex to the cluster with the closest means (center of patch).

2. **Update**: Calculate the new means to be the centroid of the vertices in the cluster.

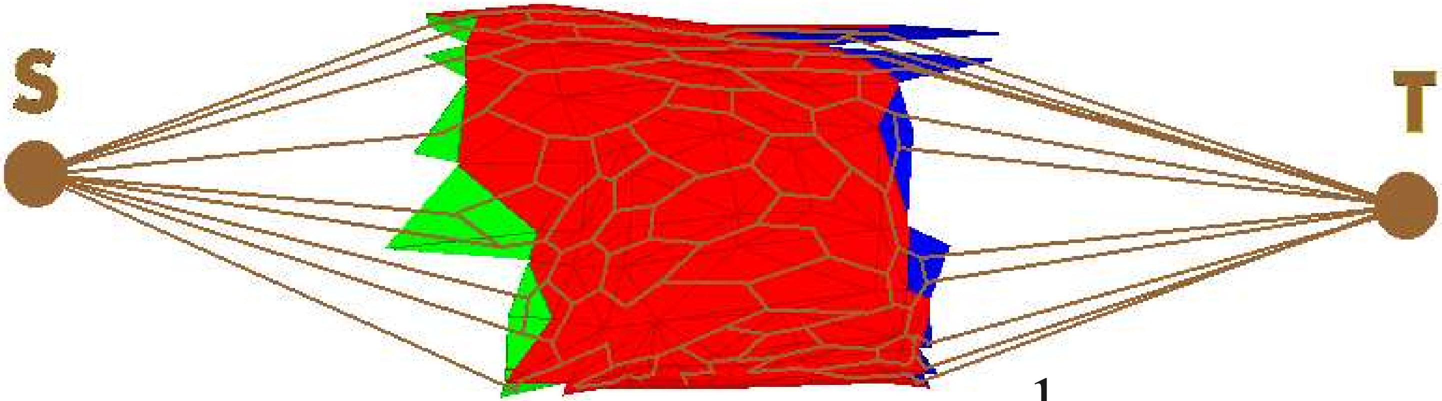
3. **Repeat** until convergence, which occurs when the assignments stop changing.



Calculate Probabilities

Probability of face f_i belonging to patch S depends on <u>relative</u> proximity of S compared to other patches

Binary case: $$P_S(f_i) = \frac{Distance(f_i, S_{seedface})}{Distance(f_i, S_{seedface}) + Distance(f_i, T_{seedface})}$$



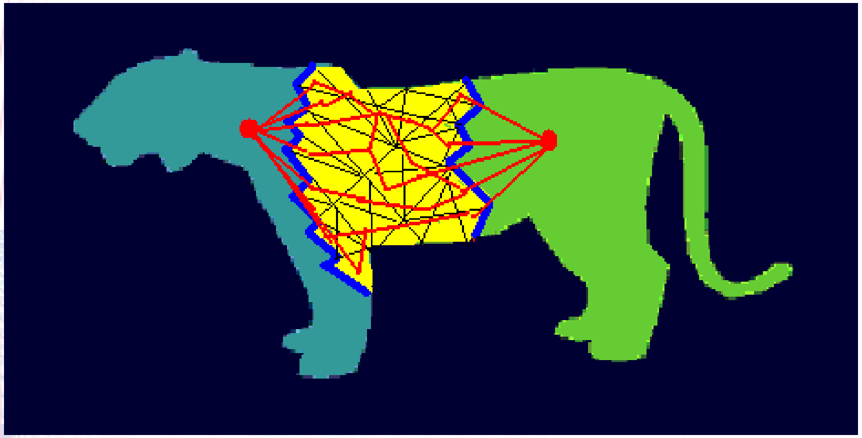
General case: $$P_S(f_i) = \frac{\frac{1}{Distance(f_i, S_{seedface})}}{\sum_{patches} \frac{1}{Distance(f_i, patch_{seedface})}}$$

Fuzzy Clustering

Fuzzy clustering considers both distance (like K-means) and probability of belonging to patch

Iteratively recompute seed faces to minimize clustering function until convergence

$$F = \sum_{p} \sum_{f} probability(f \in patch(p)) \cdot Distance(f, p)$$



We want to minimize:
- distance from face to patch,
- probability of face belonging to multiple patches

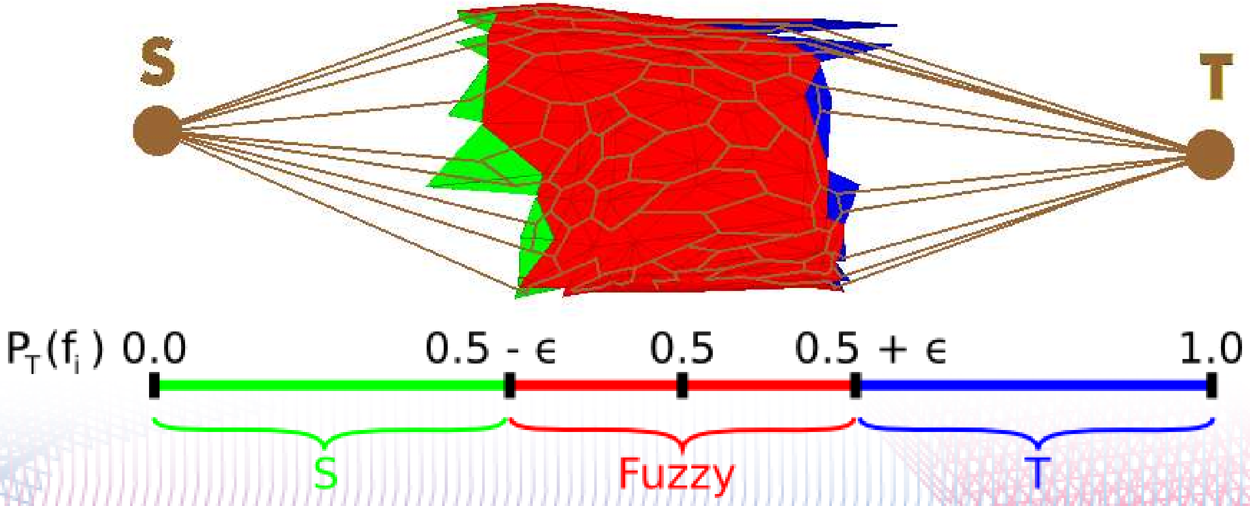
Reassigning Vertices

Choose new seed faces so other faces' probabilities of belonging to each patch are complementary

$$S_{seedface} = min_f \sum_{f_i} (1 - P_S(f_i)) \, Dist(f, f_i)$$

$$T_{seedface} = min_f \sum_{f_i} P_T(f_i) \, Dist(f, f_i)$$

Partition faces if probability of belonging to patch exceeds threshold (ϵ); remaining patches stay fuzzy



Final Decomposition

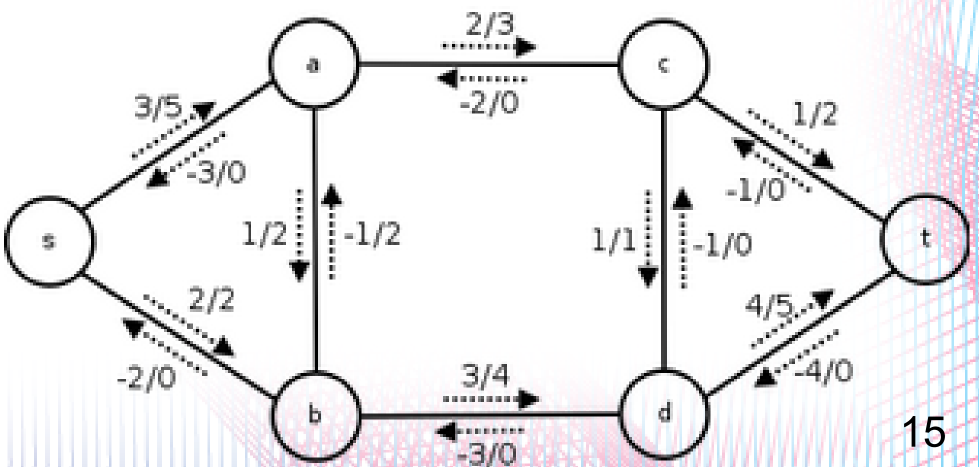
Use **network flow theory** to make the final cut

1. Flow into node = amount of flow out of node
2. Flow through $edge_{i,j} \leq$ capacity of $edge_{i,j}$

$$Capacity(edge_{ij}) = \frac{1}{1 + \frac{angularDistance(\alpha_{ij})}{averageAngularDistance}}$$

(Edges which have S,T as nodes have ∞ capacity)

Make minimum cut that passes through edges with small capacities, eg highly concave dihedral angles.



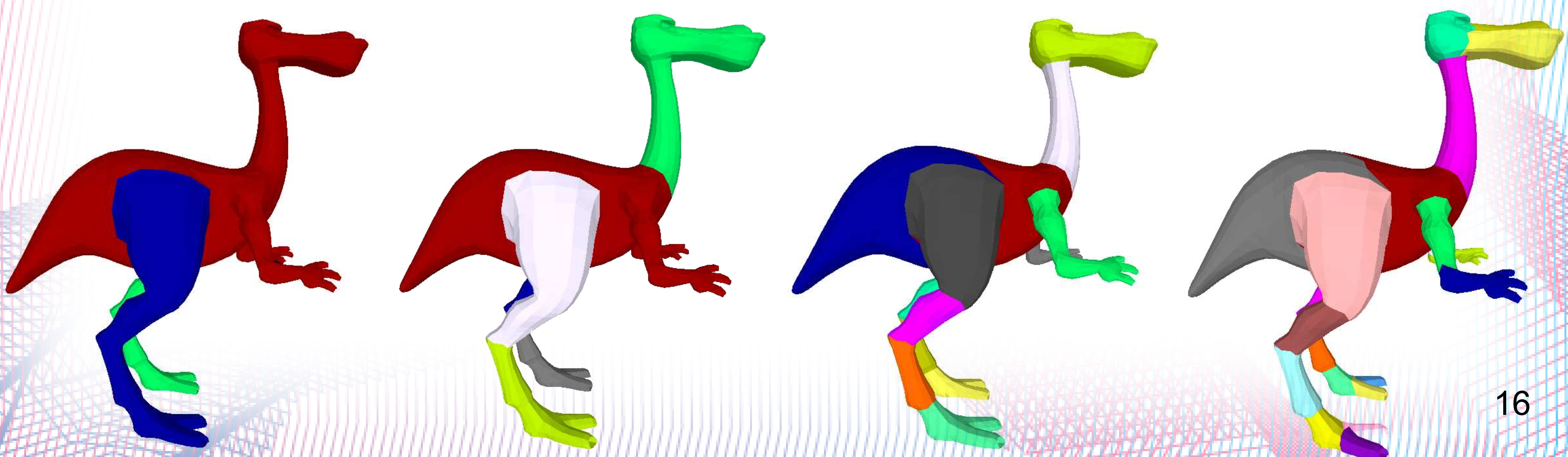
Stopping Conditions

Recursively decompose until either:

1. Distance between representatives < threshold

2. $\max(\alpha_{i,j}) - \min(\alpha_{i,j})$ < threshold (faces have similar dihedral angles → patch has fairly constant curvature)

3. averageDist(Patch)/averageDist(Object) < threshold

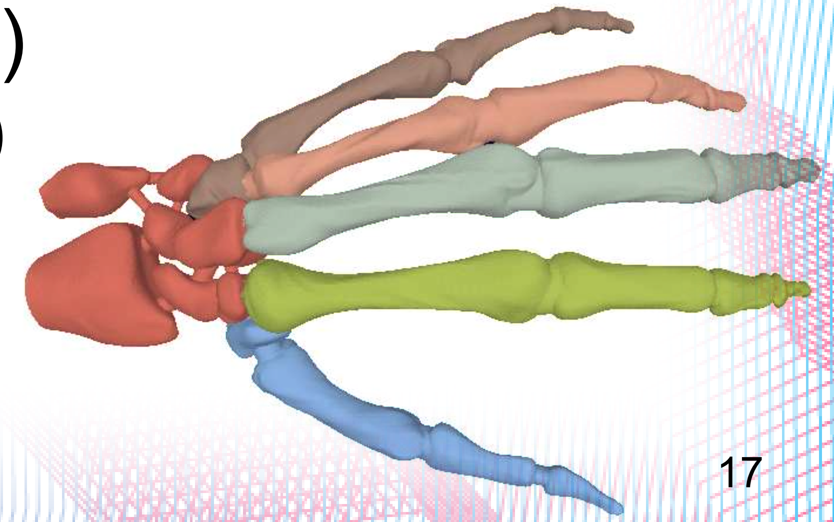


Additional Thoughts

Expedite convergence by averaging together the faces belonging to patch, rather than using specific representatives (seed faces)

Computational Complexity Considerations
V = number vertices; C = size of fuzzy region
I = number iterations in clustering algorithm

Dijkstra's Algorithm = $O(V^2 \log V)$
Assign faces to patches = $O(IV^2)$
Minimum cut = $O(C^2 \log C)$
Total = $O(V^2 \log V + IV^2)$



Possible Applications

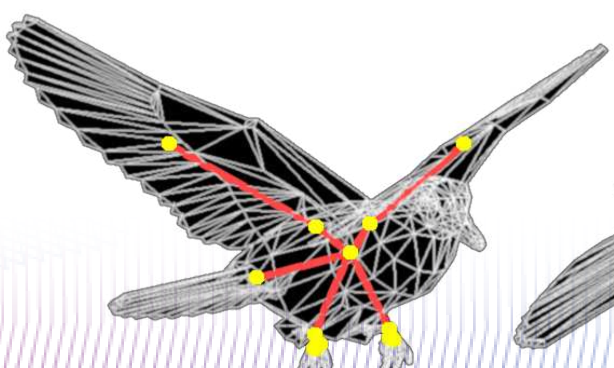
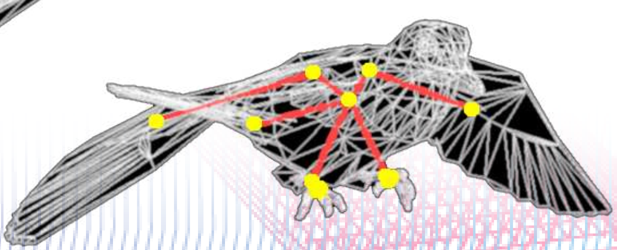
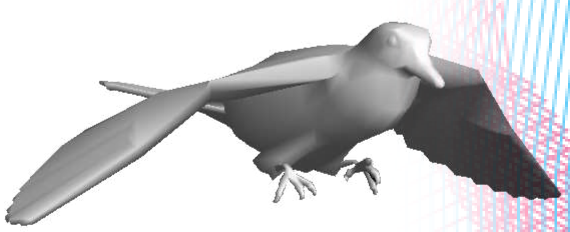
Compression – decompose mesh to simplify it

Shape retrieval – decomposition is invariant signature

Collision detection – determine bounding boxes

Texture mapping – decomposition determines parameterization of texture coordinates

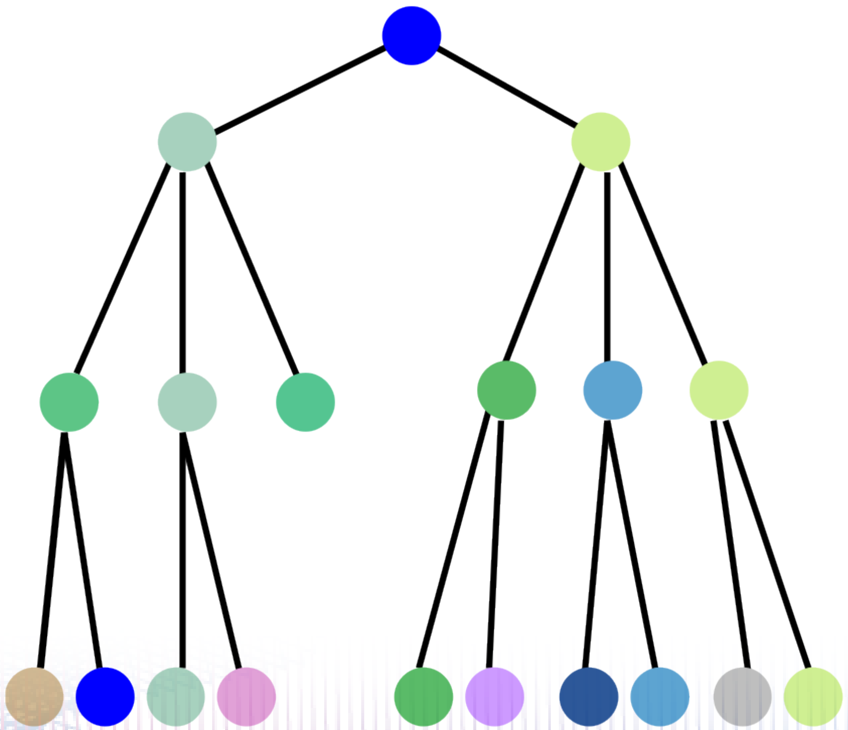
Animation

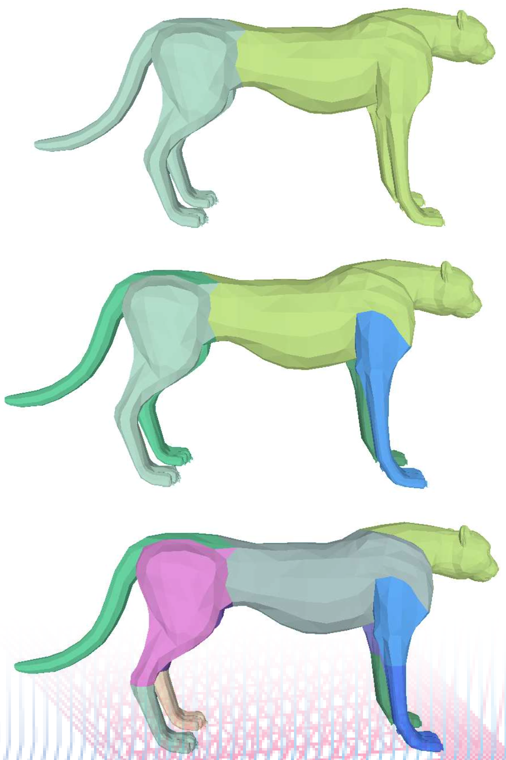





Control Skeleton Extraction

Hierarchical tree of joints for animation

1. Central patch connected to all other patches
2. Features which depend on position of other features become descendants in hierarchy
3. Joint at center of mass of each patch



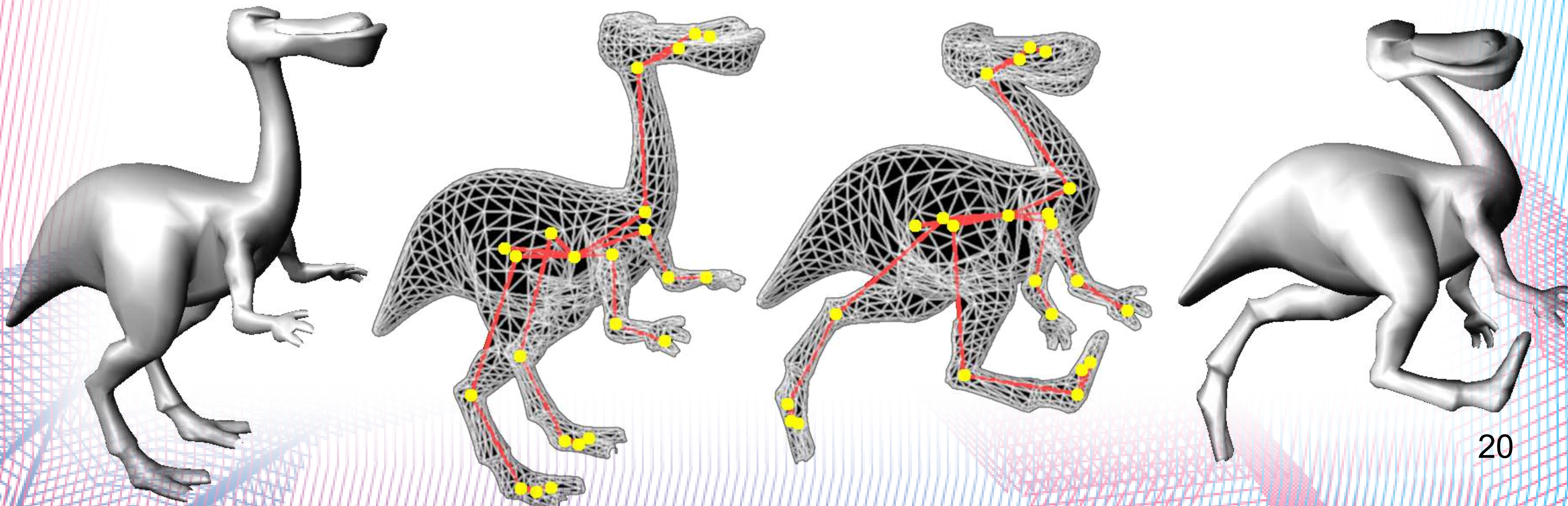


Bind and Deform Objects' Pose

Each vertex has weight, eg % faces adjacent to it that belong to a joint

Also consider cut angle, skeleton-subspace deform

Adjust joint angles to deform objects



Works Cited

Images, algorithms, and multimedia material from:

Sagi Katz, Ayellet Tal: Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts
http://webee.technion.ac.il/~ayellet/Ps/0325_ayt.pdf

Attene, Katz, Mortara, Patane, Spagnuolo, Tal
Mesh Segmentation – A Comparative Study
http://www.ima.ge.cnr.it/ima/personal/attene/PersonalPage/pdf/MeshSegm_SMI06.pdf