Inflating Balloon Models

by Yang Chen and Gérard Medioni

Overview

- Introduction
- Overview of algorithm
- In-depth explanation
- Examples
- Conclusion

Overview

- Introduction
- Overview of algorithm
- In-depth view explanation
- Examples
- Conclusion

Goal

- Reconstruct a 3D mesh from a data set
- Data set is a set of merged range images from scanners
- Surface is reconstructed using pre-generated geometry instead of using Delaunay/Voronoi

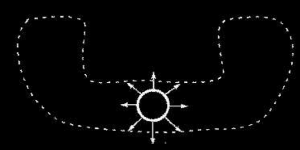
Overview

- Introduction
- Overview of algorithm
- In-depth view explanation
- Examples
- Conclusion

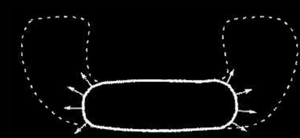
Overview of Algorithm

- Locate an area that is guaranteed to be inside the volume
- Place a icosahedron in that area such that it contains no points
- Expand/subdivide the icosahedron so that it approximates the volume

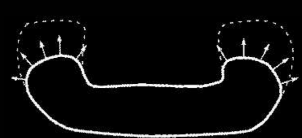
2D Example



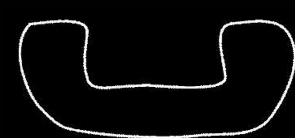
2D Example



2D Example



2D Example



Quick notes

- The algorithm starts with a small icosahedron inside the object
- Each vertex is connected to its neighbors by springs
- expand each triangle/segment along the normal based on inflation pressure inside the sphere

Overview

- Introduction
- Overview of algorithm
- In-depth explanation
- Examples
- Conclusion

In Depth Explanation

- Start off with range images
  - join images into a single point cloud
- Manually insert the starting icosahedron
  - Difficult to compute, just do it by hand
- Place the icosahedron in a queue

Calculating Forces

- Two main forces acting on each vertex
  - Inflation force pushing the vertices out
  - Spring force pulling the vertices closer together
- The spring force is calculated based upon the one-ring neighborhood of each vertex

Spring Force

- Take vertex v_i and vertices $v_{i,j}$ each of which is a neighbor of v_i
- The force applied to the vertex v_i by each vertex $v_{i,j}$ is calculated by $$s_{ij} = \frac{c_{ij} e_{ij}}{\| r_{ij} \|} r_{ij}$$
- The final force is given by summing s_{ij}, which will give the final force from all neighbor vertices
- c - spring constant, e - spring deformation, r - spring extension

Simplifications

- For more general spring motion several other parameters are used:
  - mass for inertia calculations
  - a damping value which determines the falloff of the force
- Since all of the springs/points are identical we can remove these to simplify calculations

Inflation Force

- The inflation force is calculated based upon the normal of the vertex
  - $h_i = kn_i$ where h is the inflation force on vertex v_i, k is the amplitude of the force and n_i is the normal at the vertex v_i
  - The normal at the vertex is estimated from the triangles of which v_i is a part of

Expansion Algorithm

- Create a front of the icosahedron which contains all of its faces
- Insert the front into a “front” queue
- Pop a front from the queue
- For each vertex in the front
  - calculate the spring and inflation forces
  - compute the new location of the vertex
  - compute nearest point from the dataset
  - Update the coordinates
- Discard anchored triangles
- If nTris>0 insert front into queue, repeat

Triangle Subdivision

- The expanding triangles will reconstruct the surface less accurately due to their large size
  - When expanding the spring forces between vertices become very large
  - Subdivide triangles to reduce the force
    - Creates more triangles to more closely approximate surface
    - Should create a fairly uniformly triangulated mesh

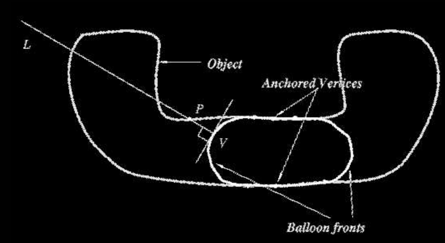
Triangle subdivision

- The triangles are subdivided so that no T-junctions exist
- Long and skinny triangles are reconnected to be wide and short

Anchoring

- A triangle becomes anchored once it reaches the surface of the point cloud
- This is determined by testing for intersection with the point set based on vertex normals
- Once a triangle is anchored it no longer moves, and its vertices are stationary

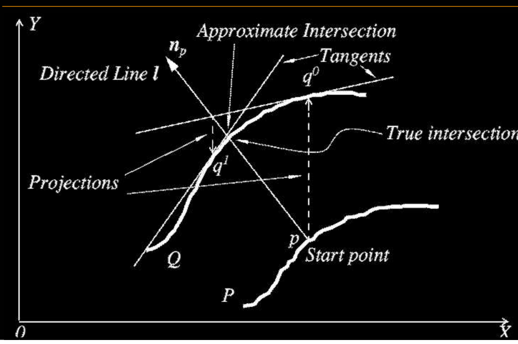
Anchoring example



Computing the intersection

- Performed by finding the intersection of a ray with a range image
- Iterative process requires refinement of the approximation
- All range scans have to be looked at to get a result

Intersection Example



Constraints

- Time step
  - Allows control over how quickly the balloon will expand
- Inflation force
  - Controls how quickly the balloon will expand
- Spring force
  - Controls the tessellation level of the triangles in the mesh

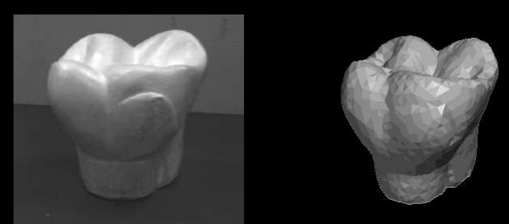
Noise + Holes

- Holes are handled similarly to the anchoring process
  - Anchor a triangle if there are no points in front
- Noise is broken down into two categories
  - Misalignment of range scans
  - Scan errors, mostly outliers
- Both of these are handled by the intersection algorithm and filtering

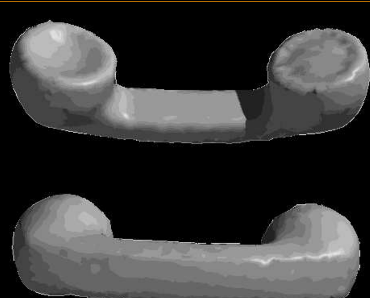
Overview

- Introduction
- Overview of algorithm
- In-depth explanation
- Examples
- Conclusion

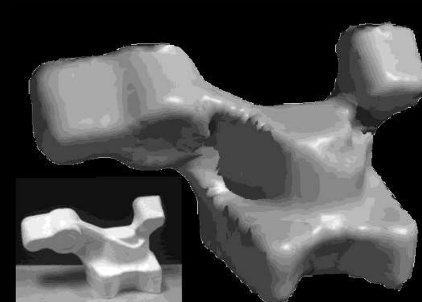
Reconstruction Samples



Reconstruction Samples



Reconstruction Samples



Overview

- Introduction
- Overview of algorithm
- In-depth explanation
- Examples
- Conclusion

Conclusion

- Presents a novel method for reconstructing 3D meshes
- Pros:
  - Guarantees a watertight mesh
- Cons:
  - Genus 0 only
  - Slow
  - Can't handle sharp edges well
  - Manual object placement