Key-component Detection on 3D Meshes using Local Features

Introduction

- Method to detect stable *components* on 3D meshes
- A component is a region on the mesh which contains *discriminative* local features
- Represent a 3D mesh with a set of regions, which they called key-components
- Used for effective matching and recognition of 3D meshes (for example 3D objects obtained by scanning, like Kinect, etc)
- Intended to be an alternative to keypoints, which are more susceptible to noise and other transformations.
- Find large and interesting structures instead of interesting points
- Important term: *Key-component*, a region on a 3D mesh with many *discriminative* local features (a subset of the mesh)

The Algorithm

1. Keypoint detection
2. Use a clustering algorithm to determine clusters of keypoints
3. Use a region growing algorithm which computes a key-component from a cluster and extracts the corresponding region on the mesh

Keypoints

A keypoint is a point on the mesh surface with a neighborhood that is geometrically unusual. For example the curvature of the surrounding mesh. Points on nearly planar surfaces are not considered interesting. This paper choose a method known as Harris 3D Method.

Reasons:

- It gives reliable results
- It is efficient
- It is easy to implement

Clustering

Key components are regions on the mesh with a high concentration of local features.

Geodesic Distance

To measure the concentration of local features, the *geodesic distance* between keypoints is used. Keypoints are grouped according to their closeness in terms of this distance.

The geodesic distance is the shortest distance between two points when you have to walk over the geometry of the surface of the mesh.

In the paper Dijkstra's Algorithm is used to approximate the geodesic distance by treating the mesh as a graph.

Clustering

There are a couple of properties that we want our clusters to fulfill:

1. Each keypoint in a cluster must be within a certain distance from at least one other keypoint in the same cluster. According to some threshold.
2. If we have two clusters, none of the keypoints in either cluster should be within a given distance from each other. According to some threshold.
3. Isolated keypoints should be discarded. Not all keypoints have to be in a cluster.
4. No keypoints should be in two clusters

Key component extraction

Starts with the set of clusters previously computed. For each cluster do the following:

Find the geodesic center of the cluster. Also, compute the smallest sphere which contains all keypoints in the cluster using Linear Programming (apparently a classic problem in computational geometry).

The geodesic center will be a point on the surface, the sphere may have a different center.

Then use a region growing algorithm using the geodesic center as the initial seed, and restricting the size of the key component using the sphere.

The region growing algorithm works as a breath first search and adds all vertices it finds that lie within the sphere to the region until all keypoints have been added.

Results

The results of the testing show that the key-components selected remain consistent for several transformations, such as noise, local scaling, holes and non-rigid transformations (movement of individual parts of the model) and there are fewer key components than keypoints by a large margin.