Algorithm Overview

Our overall algorithm for error detection can be broken up into several discrete steps. We first create a mesh of the physical object by using a depth camera. We load our JT file as a reference. The reference is then tessellated to produce a dense point cloud representing the object surface. These are then aligned using iterative closest point. Localized error is found by locating clusters of points a large distance away from the ideal point cloud.

# 3D Reconstruction from Depth Camera

The tested RealSense SDK has built in functionality to recreate a mesh from multiple captures. The produced mesh is derived from multiple depth maps captures combined into a single point cloud. This leads to the vertices in the mesh directly corresponding to depth captures of the physical objects sampled at various points.

# JT File to Mesh

Third party libraries and tools allow rendering a JT file to a mesh. This mesh represents the ideal position of vertices but does not represent surface samples. It has not been tested whether these tools generate a mesh for the exterior of the part of just inner faces. The latter will be needed for other phases or error detection. Logic may need to be added to accomplish this.

# Ideal Surface Construction

The ideal mesh can be altered to form an ideal point cloud of the object surface. New points can be added to large mesh faces by simply interpolating between the three that make the face. This allows creation of a surface cloud of arbitrary density. Preliminary testing showed the RealSense camera was generally accurate within 1cm of the physical object. This gives a lower bound for maximum distance between points.

# Alignment

These point clouds may have differences in orientation, position, and scale. The iterative closest point algorithm may be used to align our capture with the constructed ideal surface.

# Error Detection

The distance from a point in our capture to its nearest neighbor in the ideal surface cloud is a precise approximation from the distance of the point to where it should be. Mean square error may be a possible heuristic to determine overall error but does not allow finding error location. A different heuristic is used instead.

Each point in the ideal surface cloud is first added to a KD tree to allow efficient spatial queries. The nearest neighbor in the surface cloud from each capture point is used to find the distance of each point from its ideal location. There is some expected noise here. A metric such as standard deviation is used to separate noise from error. This leaves a (somewhat noisy) list of points with errors.

Clusters of points are used a heuristic in determining a defect. This can be found by putting the error points into its own KD tree and eliminating points with no close errors. A bounding box or convex hull of these points are created and the size is used to determine if an error is present.