1 Task formulation

Input: A set of points obtained from LiDAR in 3D space organized in a 2D grid $n \times m$.

Output: A set of bounding boxes that enclose simple geometric shapes (e.g. angles, spheres, cylinders, cones, etc.) that are present in the point cloud. Each bounding box is associated with a particular shape, it's parameters and a confidence value.

2 What manifold learning technique can do in this context?

Given a bounding box that contain some known shape, we can use manifold learning to determine the parameters of the shape and the confidence value. To do this, we should:

1. Extract the points that are inside the bounding box.
2. Prepare points for manifold learning: remove bias in depth, normalize the scale, etc.
3. Find k-nearest neighbors for our point in high-dimensional space.
4. Use some minimization technique to find the parameters of the shape that best fit the points.
5. Compute the confidence value of the shape from comparison of the initial points with the shape.

Important note: if after the third step our nearest neighbors are far from the query point, we can stop the process and mark the shape as not found to optimize the search.

3 How to massively parallelize the search?

We can split the grid into smaller parts and process each part in parallel. Main obstacles are:

1. We should consider different sizes of possible shapes.
2. We should not split one shape into several parts to avoid redundant computations.
3. We should consider different shapes in the same part of the grid.

Main ideas:

1. Finding k-nearest neighbors in high-dimensional space is a well-paralleled task.

2. We can iterate over bounding boxes of different sizes from the largest to the smallest and don't consider the smaller ones if the larger one is found. Besides, we may use a quadtree variation to split the grid into smaller parts.

3. If number of points in different shapes are the same, we can look for the nearest neighbors simultaneously for all shapes. And if the nearest neighbors belong to different shapes, we can assume that the shape is not found.

An important assumption: Training set has a good coverage of parameters and sufficiently dense. Otherwise, we can't guarantee that the nearest neighbors are close to the query point.