Plastic Structured Prediction Energy Network For Point Cloud Primitive Fitting

Marc Andre Piche [1] **Sreya Francis** [1]

Abstract

No current algorithm or network can learn all the possible shapes in a scene.

In this project, we take on the task of explicit representation by predicting the configuration of a graph of features with an energy network. Using both the information encoded in the vertices and edges of the graph of simple geometric features, we find what shapes arise in a point cloud. We've decided to work on point cloud as it the most difficult setting, and most methods in it can be generalized to one with more information. Unlike images, semantic learning on 3D point clouds using a deep network is challenging due to the natural way data is unstructured.

Hence we aim to do graph partitioning with the goal of finding the lowest cost unions, but where the result of alterations is unknown unless we compute the energy. But also with the possibility of edge additions. In a sense like a flow of mixed elements: if we let a node flow into a set, that element may "react" to increase or dampen the energy. Finally, we want unsupervised learning as most use cases have no ground truth; we simply want the best solution. We hope the network architecture presented here will the reader's interest as much it did ours.

1. Goal and Motivation

With this project, we aim at taking a significant step towards explicit shape representation wherein it is possible to define structures of generally more refined shapes. Our aim is to predict a shape explicitly, a challenging endeavour as contrary to intuition, there is no class to a shape. A class has may have a shape, but a shape has no class. And therefore we are in the situation of a regression task where multiple instances taking any form are to be predicted.

[*]Equal contribution [1]Department of Computer Science, University of Montreal, Montreal, Canada. Correspondence to: Marc Andre Piche <marc-andre.piche@umontreal.ca>, Sreya Francis <sreya.francis@umontreal.ca>.

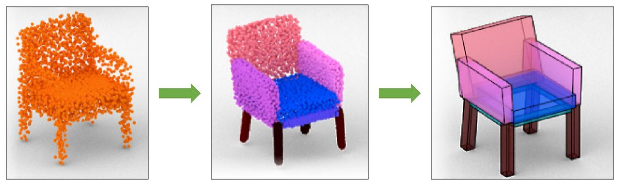

Figure 1. Decomposition of point cloud to composing shapes.

2. Previous Work

Our work is inspired from the field of geometric fitting. Most of the existing algorithms in this field classify a point just based on the type of surface and hence work only with planes and cylinders. There has been a lot of research into solving this task, with RANSAC (5)(6) still being the reference. Most previous work also just identify membership of a point to a primitive class considering its neighbours or simply create a convex hull. But our aim goes beyond that. We want to explicitly represent an object. And since the entire computer graphics field is already doing that; with meshes, curves, surfaces and so on. We want to output a prediction compatible with these structures. But having a learning algorithm output this is not trivial. Some progress has been made here in recent years, but the methods are still very limited. Most of the existing algorithms in this field classify a point just based on the type of surface and hence work only with planes and cylinders.

Hierarchical decomposition: "Defining a structure of generally ever more refined shapes". Let's clarify what we mean by searching towards explicit representation. As we believe this type of machine learning progress would. We would like to encode a shape in a hierarchical shape decomposition. A tree of the parts defined by their subparts, from the global shape to its details. It is not a new idea as it goes back to Marr's level of representation (D. Marr 1982). But it is very difficult to actually accomplish , it would have more impact beyond having trivial classification (as seen in figure

2 hand), but would allow for machines to communicate with us what they see or think. But more importantly would allow for machines to communicate shapes between themselves directly. Which is difficult even for humans to do when we think about it. How do you do describe the shape of a rose to someone who has never seen a flower?

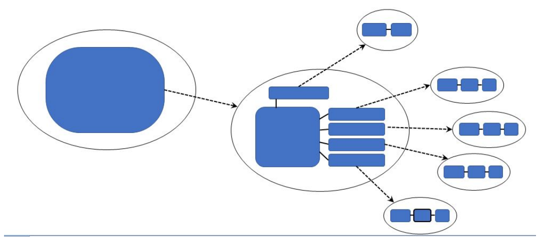

Figure 2. Shape decomposition

3. PointNet(1) and our Leaf Node Encoder

Predicting the optimal shape to fit a point cloud is best seen as a regression task. And this is how we approached the first part of our problem. The leaves of our segmentation tree are the shapes to output. For this we project a cloud onto a predicted shape (figure 3). We used PointNet (1) as an encoder to accomplish this task in self-supervised learning by predicting the parameters of a randomly generated shape. PointNet(1) is a simple neural network with one neuron per point and including Spatial Transformer Network (7) between layer blocks.

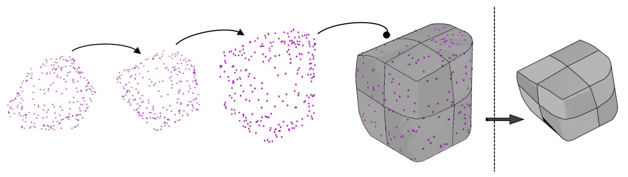

Figure 3. Shape Decomposition

We also included a curriculum approach to make sure the networks learns more important shapes thoroughly. To do that, our dataloader generates samples by varying along a Dirichelet distribution to increase the eccentricity over time. The network converges very rapidly as expected except for rotation. Rotations of more than a few degrees cause us problems we could not aside from doing iterative as in (1) which we would like to avoid.

3.1. Shooting Method

Instead of raytracing, to generate our point cloud we sample on the surface of the object using a Poisson distribution. Using a kd-tree, we then select a point randomly to remove it along with 30% of the cloud nearest to this point to simulate occlusion.

4. Octree Hausdorff Distance

We explored many distances as our cost function and settled on Hausdorff as it is defined to represent the distance between two sets. It was also accessible in the IGL(9) library. a synthetic point cloud simplification method to obtain computationally manageable point sets.

The Hausdorff distance measures how far two subsets of a metric space are from each other. Given two finite point sets A= {a1, .. an} and B= {b1 ... bm}, the Hausdorff distance is defined as:

$$H(A,B) = max(h(A,B), h(B,A))$$
$$h(A,B) = max_{a \in A} min_{b \in B} \mid A - B \mid$$

where the Hausdorff distance H(A, B) is the maximum of h(A, B) and h(B, A), the function h(A, B) is called the directed Hausdorff distance from A to B.

First, a coarse-to-fine feature extraction manner is designed with normal vectors deviation and k-means clustering methods, which can concentrate more sample points in regions of high curvature. Additionally, the directed Hausdorff distance is performed directly on the point cloud which samples the point cloud judiciously with an edge-preserving manner. As is it very sensitive to outliers by its definition, we modified this one to a partial Hausdorff's distance. Where the 75th percentile point is considered the farthest point.

5. Primitive Block

The first step in our project was defining what structure we exactly want to predict. Considering at first each primitive as its own class proved to have a limited scope. We realized there is a good reason most algorithms only fit planes and cylinders, as for example, cones and ellipsoids are really hard. We call what we use as a building block for predicting all shapes a Primitive Block (figure 4). It cannot, of course, predict all types of shapes, but a good portion of convex shapes is a good start. To encode this flexible primitive we took inspiration from the quad-edge datastrucure where we have links to vertices, edges and faces.

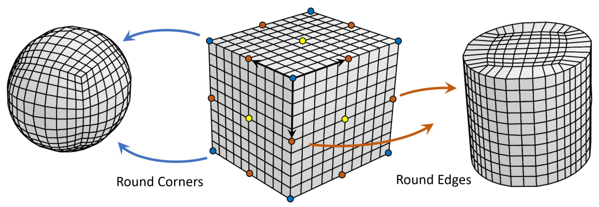

Figure 4. PrimBlock

We created a "cube" with arbitrary smoothness that has each of its octants defined by the roundness of either the corners or an edge. (It is exclusive as described : we tried allowing for a combination but our structure cannot support it). This structure also has the benefit of being similar to an octree allowing for fast computation of distances. By considering transformations in our parameters expands our Primitive Block even more.

6. Point Cloud Challenges

Point clouds are very complicated when compared to images. It is almost impossible to tell a shape with a handful of coordinates which are unevenly distributed as well as unsorted as they can be anywhere in space. The sparseness of point cloud in 3D space makes most spatial operators inefficient. Moreover, the relationship between points is implicit and difficult to be represented due to the unordered and unstructured property of point cloud. The best we can do is build a graph of neighbours which comes with a cost of being very expensive.

We have listed below some major challenges faced:

- Unordered and unstructured list
- Sparsity (challenge for CNN)
- Density variability
- Occlusion
- Huge data size

Several lines of solutions have been proposed to address these problems such as Voxels, 2D projection with 2D convolution, 3D convolution (expensive), Directly work on cloud as in PointNet(1) etc , each with their own drawback.

7. Graph Partitioning or Clustering or Segmentation by features

For now with, we consider just convex shapes and try to formulate this as a combinatorial matching problem with reduced cost. There are many equivalent ways this name this task, but they are not equal. We've mentioned this earlier, but it is the most important thing to note, at least in our opinion as it's a major hurdle in our problem setting; is that we cannot learn classes and segment using class labels. Even if we could define a sort of single "continues class" where say you could differentiate 2 objects by their shades of colour, 2 objets that exactly the same side by side who be considered a single object here. It would allow us to use one of the many methods for image segmentation, and we couldn't find a way around this problem. The solution that emerged thanks to the SuperPoint (3) algorithm is to instead predict a graph of relations and do clustering with these. That formulation was the biggest breakthrough in our project as it is semantically richer than all our previous attempts.

We predict a graph such that, simply, the geometric features that are connected belong to the same object.

We follow the global energy model for computational efficiency. Another advantage is that the segmentation is adaptive to the local geometric complexity.

Let us consider the input point cloud C as a set of n 3D points. Each point i $\in$ C is defined by its 3D position, and, if available, other observations such as colour or intensity. For each point, we compute a set of d_g geometric features $f_i \in R^{d_g}$ characterizing the shape of its local neighbourhood. We use three-dimensionality values : linearity, planarity and scattering, as well as the verticality feature . We also compute the elevation of each point, defined as the z coordinate of pi normalized over the whole input cloud. The global energy is defined with respect to the 10-nearest neighbour adjacency graph G_{nn} = (C, E_{nn}) of the point cloud. The geometrically homogeneous partition is defined as the constant connected components of the solution of the following optimization problem:

$$argmin_{g\in R^{d_g}} \sum_{i\in C} \mid g_i - f_i \mid^2 + \mu \sum_{(i,j)\in E_n n} w_{i,j}[g_i - g_j] \neq 0$$

where [·] is the Iverson bracket. The edge weight w $\in R_+^{|E|}$ is chosen to be linearly decreasing with respect to the edge length. The factor μ is the regularization strength and determines the coarseness of the resulting partition.

8. Our SPEN formulation

Our SPEN formulation (figure 5) comes down to 5 steps :

- Produce a graph of SuperPoints
- Encode each SuperPoint π such that we can predict how well they match pairwise
- Compute their matching score
- Label the SuperPoints using edge clustering
- Predict the shape for each object ω composed of the superpoints with the same label.

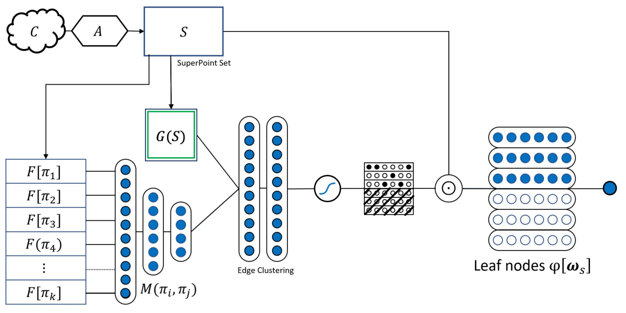

Figure 5. Plastic SPEN Network.

8.1. Matching pairs

It is an expectation density we would ideally like to learn. (figure 6) Attention to certain regions of space. It is clear that some sort of neural network would be ideal for this task, as it what our brain does all the time. For example, by completing shapes or considering symmetries as part of the same object. We spend some time trying define such a model, but set it aside for now and it's a goal we shouldn't forget if anyone reading has a clever inspiration in the future.

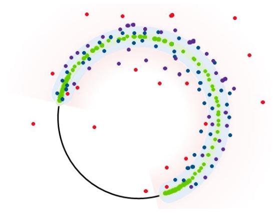

Figure 6. Learn expectation density via attention to specific regions of space.

What we settled for instead is a matching score between every pair in the graph. That we can think as a correlation between 2 Superpoints. How are these 2 Superpoints likely to fit together ? This can be trained easily again in a self-supervised way by predicting the impact on error from the union of two SuperPoints.

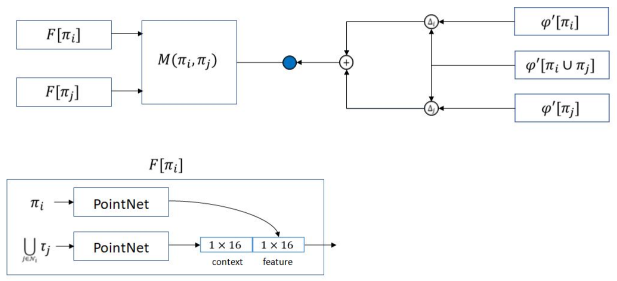

Figure 7. Calculation of Matching Score between every pair in the graph.

We know that basic SuperPoint information from which it was chosen can be encoded in 12D (4 vectors). Here we arbitrarily let the encoder choose its own representation with a 16D encoding.

8.2. End-to-end learning for structured prediction energy networks

This forms the perfect feature network for our SPEN formulation as it encodes the feature edges of our graph. By throwing away the prediction, we output exactly how 2 parts that fit together.

8.3. Edge Clustering

This where, we think, our problem of learning something without classes caught up to us again. We are not sure this is the reason. We have an adjacency matrix, and our edge features available to predict labelling based on clustering. I should be the perfect information for this task. A simple algorithm (we want ours to be differentiable) could do it based on the adjacency matrix alone. Yet a MLP or CNN proved ill-suited for this task. We have to study graph neural networks, but our hope is that there is a simple solution as it the last missing to complete our SPEN. What we are trying now, is to do Edge Clustering as we've imagined it in figure 8. With one or more layers defines as those, where edge features of a node are connected to along with the neigbourgh's label, resulting in a random field that predicts its own clustering.

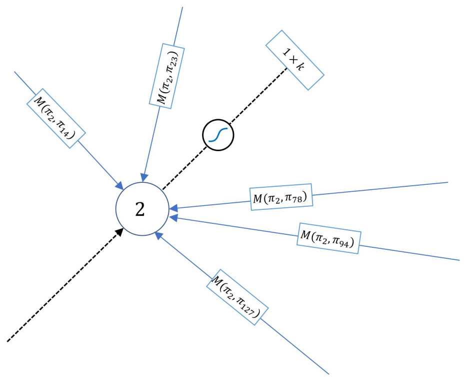

Figure 8. Edge Clustering Network

8.4. Energy function

With this labelling, we can select all SuperPoint π_i belonging to the same object and send these points into our Leaf Node. The leaf computes the error on each π_i separately to return the gradient. We've had to code this in an expensive way as "manual gradient" where we compute the error for a ϵ step for parameters (there are 51) for every $\pi_i \in \omega_j$

9. Implementation details

Our implementation could be found at:

https://github.com/sreyafrancis/PlasticNet

- PrimBlock

  Our Primblock datastructure consists of a QuadEdge structure that can both represent faces and edges wherein corner values denote Spheriphicity and edge values denote Cylindricity.This implementation provides us many advantages one among which is the very fast computation of distance to a quad as this primblock data-structure is very similar to an Octree.

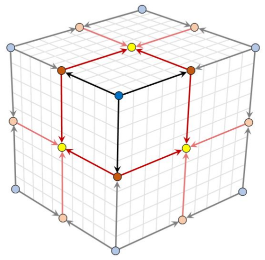

Figure 9. PrimBlock

- Hausdroff

  Our leaf loss function includes one way Ranked Haursdoff with Convex hull penalty. One of the major issues with hausdroff distance is that Hausdorff outlier problem occurs with transformations (especially rotations). We used averaged/partial Hausdorff so as to bring down this error.Large affine "squashing" can have a big impact on precision.

- PointNet(1)

  We use pointnet(1) as it directly consumes point clouds and well respects the permutation invariance of points in the input.This provides a unified architecture for applications ranging from object classification, part segmentation, to scene semantic parsing. It helps us in mapping towards an original Primitive.

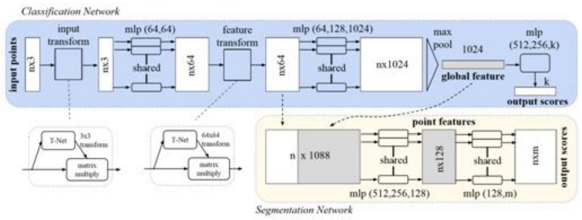

Figure 10. Pointnet(1) network architecture.

- Capture Origin

  We also face an issue that we can call global dual solutions. That we can address in a way that gives us an Opportunity to skip computation even further and get better results. Simply by considering : Are we inside our outside the object. "Well, where is the origin ?" Meaning we can check the validity of an edge with the space it traverses.

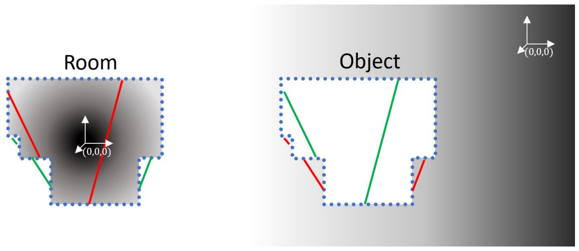

Figure 11. Capture origin.

In a large scene, this would eliminate most edges. There are many ways to compute this space, like marching cubes or octree decomposition. We did not implement this part yet, but will be included in the future.

9.1. Yet to implement

There is still a lot of pointers to be considered to be implemented in our project:

- SPEN : Our SPEN is composed of 4 sub-networks. And to complete our network, like we said earlier, we need a good labelling network that can take full advantage of all the rich information we provide it. But for a full End-To-End SPEN we also need to compute the unrolling gradient over every iteration, which we haven't started into yet.
- Data : Right now, we use our synthetic dataloaer, and it can only produce shapes the network was trained to predict. We can't wait to try algorithms on PartNet (8) and then on real data to see how we will fare on real data with noise and occlusions.
- Mesh replacement : Finally, our algorithm will truly be complete and usable when it will be able to replace parts of a point cloud with our primitive blocks.
- Expectation Density : We would also like to learn the expectation density via attention to specific regions of space so as to make our algorithm even more complete.

References

[1] Charles R. Qi, Hao Su, Kaichun Mo, Leonidas J. Guibas. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. CVPR, 2018.

[2] David Belanger, Bishan Yang, Andrew McCallum. *End-to-End Learning for Structured Prediction Energy Networks*. ICML, 2017.

[3] Loic Landrieu, Martin Simonovsky. *Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs*. CVPR, 2018.

[4] Loic Landrieu, Mohamed Boussaha. *Point Cloud Oversegmentation with Graph-Structured Deep Metric Learning*. CVPR, 2018.

[5] M.A. Fischler, R.C. Bolles. *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography*. Communications of the ACM, 24(6):381–395, 1981.

[6] Ruwen Schnabel, Roland WahlRol, KleinReinhard Klein. *Efficient RANSAC for point-cloud shape detection*. Computer Graphics Forum 26(2):214-226, 2007.

[7] Max Jaderberg, Karen Simonyan, Andrew Zisserman, Koray Kavukcuoglu. *Spatial Transformer Networks*. CVPR, 2016.

[8] Fenggen Yu, Kun Liu, Yan Zhang, Chenyang Zhu, Kai Xu. *PartNet: A Recursive Part Decomposition Network for Fine-grained and Hierarchical Shape Segmentation*. CVPR, 2019.

[9] IGL library: `https://libigl.github.io/`.