Title:
Encoding Partial Point Cloud Neighborhoods for Convolutional Neural Networks

Authors:
Tathagata Chakraborty, tathagata.chakr@hcl.com, HCL Technologies
Hariharan Krishnamurthy, hariharan_k@hcl.com, HCL Technologies

Keywords:
Convolutional Neural Networks, Point Cloud, Hole Patching, Mesh Completion

DOI: 10.14733/cadconfP.2022.61-65

Introduction:

The remarkable success of convolutional neural networks (CNNs) on long standing problems in computer vision has led to the recent resurgence of interest in deep neural networks and their applications in other domains. CNNs can progressively learn discriminative hierarchical features, thus capturing the underlying structure of the data very effectively, and giving state-of-the-art results on problems in many domains.

It is however difficult to extend CNNs to 3D data. CNNs have been applied to rendered 2D projections of 3D data for 3D object classification [7]. 3D CNNs have been used with voxelized 3D data and octrees for shape analysis and retrieval [9]. CNNs have also been trained on small patches sampled from a mesh surface and used for in-painting meshes [6]. Points from 3D point clouds have been embedded into high-dimensional spaces [5] and high-dimensional feature descriptors of mesh vertices have been learned using graph-based methods [3][4]. However, the lack of a natural orientation and ordering of point data in 3D thwart efforts towards effectively leveraging these deep learning-based techniques. For a comprehensive survey of different methods for encoding 3D data for use with deep neural networks see [1].

In this paper we describe a simple and easy to implement method for discretely encoding partial point neighborhoods for direct input to traditional CNNs. Our encoding method captures and preserves most of the structural information present in a point neighborhood. We show the usefulness of this technique for accurately predicting point coordinates and other properties. In particular, we train a CNN model with our encoding and show its effectiveness in patching holes in meshes.

Background:

The use of deep learning methods in 3D have been impeded by issues related to the nature of 3D CAD data. One-dimensional time-series data and 2D image data both have strong structural properties and natural orientations that can be leveraged by CNNs and recurrent neural networks (RNNs). On the other extreme, with high-dimensional data, we hope that the underlying structure, is discovered automatically by a deep enough neural network. 3D CAD data sits uncomfortably on the boundary between these.

3D mesh and point clouds have enough structure that it seems important to use neural network models like CNNs that explicitly leverage this underlying structure without having to learn it from a large number of samples. However, there are enough degrees of freedom in 3D, particularly in how the 3D points are oriented and ordered, that it becomes difficult to normalize the data for use with such

networks. Completely abandoning the structure of the 3D data, on the other hand, restricts its usefulness to global inference tasks such as object recognition and classification.

Every approach to deep learning in 3D has its own entailing issues. 3D CNNs on voxelized data are both memory and compute intensive. Voxelization also introduces surface inaccuracies thus precluding accurate prediction of local surface properties. Techniques where the point data is embedded in a high-dimensional space and then aggregated for pose and order invariance also cannot be easily extended to local estimation tasks. Projection and patch-based methods are not always very accurate, and graph-based methods haven't yet been investigated for local interpolation tasks.

Our method is inspired and integrates ideas from graph-based [3][4][2] and patch-based methods [6]. However, our method works directly with untransformed point clouds and is comparatively simple to implement. Our method neither requires creating a height map as in patch-based methods, and nor does it require conversion to polar coordinates or any special neural network machinery as in many graph-based methods. We encode the point neighborhood as a layered 2D image containing raw coordinates and properties which can be directly used to train a standard CNN.

We illustrate our method in the context of patching holes on smooth surfaces, but the method can be used to predict and interpolate other properties of point clouds and meshes. Our broad approach for patching holes is based on the advancing front method described in [8], but instead of using moving least squares (MLS) as in [8] we estimate the points using a CNN.

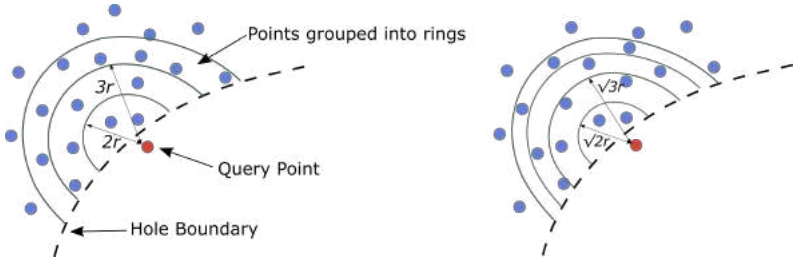

Fig. 1: Left: Neighborhood points grouped into rings with linearly increasing radii; Right: Non-linearly increasing ring radii for enclosing an equal number of points within each ring annulus.

Encoding Partial Neighborhoods:

We describe a method for discrete encoding of partial point cloud neighborhoods. The encoding method can work with other 3D formats. Points clouds however represent a common denominator for 3D data, and are also convenient for illustrating our method. We therefore assume that the support faces surrounding the hole boundary can easily be approximated by a uniformly distributed point cloud and that such an approximation is available in a data structure where nearest neighbor queries can also be performed efficiently.

In an advancing front technique, the hole is patched by progressively interpolating from the boundary and towards the interior of the hole. The neighborhood of a query point on the advancing front of the hole boundary then typically looks like that shown in Fig. 1, where the query point is colored red, the neighboring points are colored purple, and the hole boundary indicated with the dashed curve. As the advancing front grows and the hole correspondingly contracts in size the neighboring points tend to surround the query point more completely on all sides.

For encoding a partial neighborhood, we first identify the points in a sufficiently large neighborhood of the query point. These neighboring points are then grouped into several rings based on their distance from the query point (see Fig. 1). In our experiments the consecutive ring radii are linearly increasing

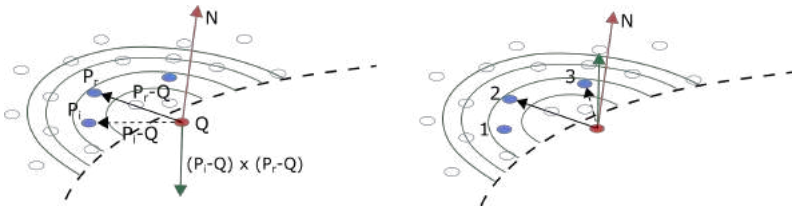

Fig. 2: Left: Point distance computation which gives a negative value; Right: Positive point distance result, also showing the sorted point order for the points in the ring.

multiples of the point cloud resolution r (that is, r is the mean distance between two neighboring points in the point cloud). However, for other problems the ring radii may be differently determined.

The point encodings described in [3][4][2] are not rotation invariant. In general, it is impossible to uniquely order a 360° neighborhood around a point. However, partial neighborhoods can be ordered consistently. We first select a random point P_r from the ring. Then for each of the rest of the points P_i in the ring, we compute the cross product $C_i = (P_i - Q) \times (P_r - Q)$. Then the dot product $d_i = (P_i - Q) \cdot (P_r - Q) \times sign(N \cdot C_i)$ gives us the signed distance of the rest of the points P_i from the point P_r based on which the points inside a particular ring can be ordered. Note that here Q is an estimation of the query point and the N the estimated normal at this point. The computation is visually illustrated in Fig. 2.

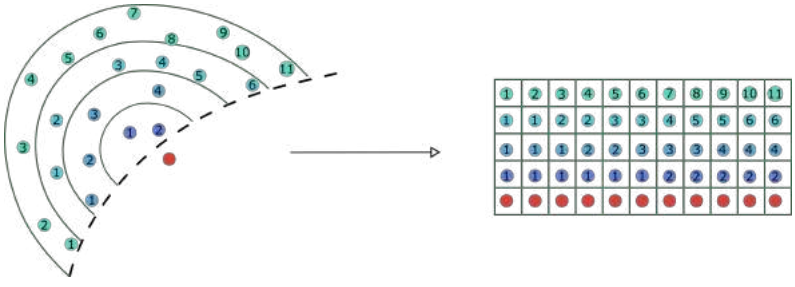

Fig. 3: Left: Ordered points in each ring; Right: Ordered points encoded as an image by oversampling the points to fill the columns in the image.

With the points in each ring ordered, we next oversample the data in each ring into fixed sized vectors which are then stacked to form the rows of a 2D image. The 2D image has one more row than the number of rings, each of the top rows corresponding to each ring with an additional bottom row for the query point. The number of columns in the image is set to the number of points in the outermost ring, although a larger or a smaller number can be used depending on the resolution of the point cloud. Each of the smaller inner rings will therefore contain fewer points than are required to fill the image rows. The points in the rings must therefore be oversampled while preserving their order to artificially stretch the data. This process is illustrated in Fig. 3.

Oversampling the ordered point data in the partial neighborhoods to create 2D images allows us to directly use standard CNN architectures and models without having to invent special convolutional kernels and other neural network machinery. Such a 2D image representation also preserves much of the structure present in both the circular and radial directions around the query point. The only structural information loss is due to the nature of the point distribution inherent in the point cloud approximation of the neighborhood which in many cases may not be uniform with respect to our discretization scheme.

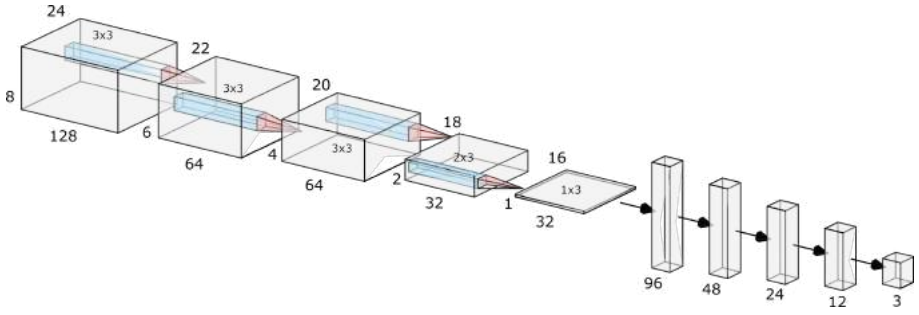

Fig. 4: CNN network architecture for 2D images of size 8×24

The exact data that should be stored in these 2D images will depend on the type of point property to be estimated. Here some creativity will often be required and some inductive bias must necessarily creep in before the CNN model can learn from the data and interpolate well. We tried several input data configurations and combinations before we found the set of data that works reasonably well for interpolating point coordinates. Instead of directly predicting point coordinates, we predict the point normal and pivot the query point to align its normal with the predicted normal. As input to the network we use the point normals and the relative point coordinates of the neighborhood points, in addition to the absolute point coordinates.

CNN model architecture:

Once the input data has been converted to 2D images standard CNN model architectures can be used for training and prediction. For hole patching we use a simple CNN with 5 convolutional layers with 3×3 filters followed by 5 fully connected layers. Our input is a set of 9-layered 2D images, each 8×24 in size ($7 + 1 = 8$ rings each containing 24 point data). The small size of the input image limits the number of convolution layers we can use without extra padding in each convolution step. The architecture we use is illustrated in the Fig. 4. Note the lack of any max-pooling or normalization layers since here our 8×24 images are possibly too small to derive any benefits from such layers.

Results:

Fig. 5 show some results of patching meshes using our CNN model. One can see slight undulations on the patched surface which can be smoothened out in a post-processing step. The result on the right shows a hole spanning a faceted surface where the connecting edges have been filleted. These cases can't easily be patched using traditional methods that locally fit smooth surfaces. A CNN model, on the other hand, can learn much larger regions of the model during training, and can memorize the varying underlying structure of a non-smooth surface. In fact, the more faceted and noisy the surface the better the CNN model is likely to perform compared to traditional methods which are designed mainly for patching smooth surfaces. This suggests that traditional and deep learning-based techniques are complementary in nature and could possibly be used together for patching different regions of a large complicated mesh, like those acquired from 3D scanning methods.

Conclusions:

In this paper we describe a simple and easy to implement method for encoding partial point neighborhoods for use with CNNs and other deep learning methods. The encoding allows us to capture the local structural information without the loss incurred due to aggregation required for orientation and order invariance in other techniques. Instead of full rectangular or circular patches we suggest sampling partial patches

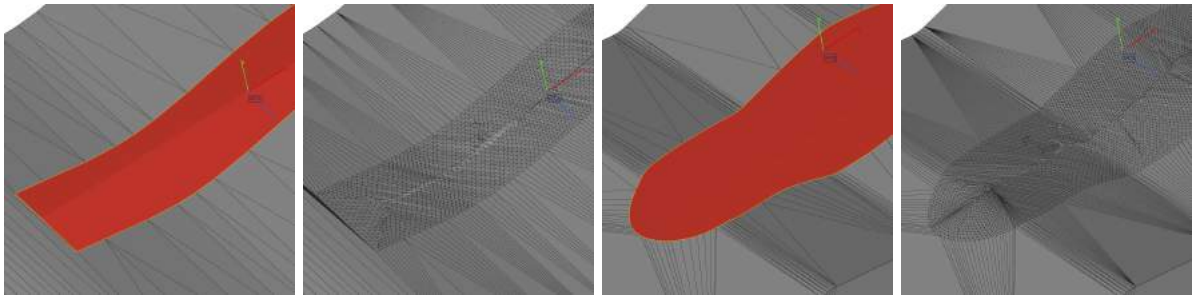

Fig. 5: Some hole patching results using CNNs. The first image shows the mesh with the hole and the second one shows the patched result.

from the mesh to train a neural network model. Choosing a partial neighborhood allows us to order the points in the neighborhood consistently. The method is designed to predict and interpolate local mesh properties, and can be adapted for use with different deep learning models for other tasks.

References:

[1] Ahmed, E.; Saint, A.; Shabayek, A.E.R.; Cherenkova, K.; Das, R.; Gusev, G.; Aouada, D.; Ottersten, B.: A survey on deep learning advances on different 3D data representations, arXiv preprint arXiv:1808.01462, 2018.

[2] Lim, I.; Dielen, A.; Campen, M.; Kobbelt, L.: A simple approach to intrinsic correspondence learning on unstructured 3d meshes, In Proceedings of the European Conference on Computer Vision (ECCV) Workshops, 2018. https://doi.org/10.1007/978-3-030-11015-4_26

[3] Masci, J.; Boscaini, D.; Bronstein, M.; Vandergheynst, P.: Geodesic convolutional neural networks on riemannian manifolds, In Proceedings of the IEEE international conference on computer vision workshops, 2015, 37-45. https://doi.org/10.1109/ICCVW.2015.112

[4] Monti, F.; Boscaini, D.; Masci, J.; Rodola, E.; Svoboda, J.; Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns, In Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, 5115-5124. https://doi.org/10.1109/CVPR.2017.576

[5] Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation, In Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, 652-660. https://doi.org/10.1109/CVPR.2017.16

[6] Sarkar, K.; Varanasi, K.; Stricker, D.: Learning quadrangulated patches for 3d shape parameterization and completion, In 2017 International Conference on 3D Vision (3DV), 2017, October, 383-392. IEEE. https://doi.org/10.1109/3DV.2017.00051

[7] Su, H.; Maji, S.; Kalogerakis, E.; Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition, In Proceedings of the IEEE international conference on computer vision, 2015, 945-953. https://doi.org/10.1109/ICCV.2015.114

[8] Tekumalla, L.S; Cohen, E.: A hole-filling algorithm for triangular meshes, School of Computing, University of Utah, UUCS-04-019, UT, USA, 2, 2004.

[9] Wang, P.S.; Liu, Y.; Guo, Y.X.; Sun, C.Y.; Tong, X.: O-cnn: Octree-based convolutional neural networks for 3d shape analysis, ACM Transactions On Graphics (TOG), 36(4), 2017, 1-11. https://doi.org/10.1145/3072959.3073608