DESIGN AND ANALYSIS OF ALGORITHMS
Paper: Neural Mesh Flow: 3D Manifold Mesh Generation via Diffeomorphic Flows
(Analysis of Code)

Link: https://paperswithcode.com/paper/neural-mesh-flow-3d-manifold-meshcode

Bisma Ijaz, 351995
Imaan Imtiaz, 331780

June 2, 2022

1 Summary

Meshes are important representations of physical 3D entities in the virtual world. Applications like rendering, simulations and 3D printing require meshes to be manifold so that they can interact with the world like the real objects they represent. Prior methods generate meshes with great geometric accuracy but poor manifoldness. In this work Neural Mesh Flow (NMF) was proposed to generate two-manifold meshes for genus-0 shapes. Specifically, NMF is a shape auto-encoder consisting of several Neural Ordinary Differential Equation NODE blocks that learn accurate mesh geometry by progressively deforming a spherical mesh. Training NMF is simpler compared to state-of-the-art methods since it does not require any explicit mesh-based regularization. The experiments in the paper demonstrate that NMF faciliate several applications such as single-view mesh reconstruction, global shape parameterization, texture mapping, shape deformation and correspondence. Importantly, it is demonstrated that manifold meshes generated using NMF are better-suited for physically-based rendering and simulation. In this document we analyse the algorithms that have been used in this paper.

2 Marching cube algorithm

Marching cubes is a computer graphics algorithm, published for extracting a polygonal mesh of an isosurface from a three-dimensional discrete scalar field (the elements of which are sometimes called voxels). The applications of this algorithm are mainly concerned with medical visualizations such as CT and MRI scan data images, and special effects or 3-D modelling with what is usually called metaballs or other metasurfaces. The marching cubes algorithm is meant to be used for 3-D, the 2-D version of this algorithm is called the marching squares algorithm.It is a simple algorithm for creating a triangle mesh from an implicit function (one of the form f(x, y, z) = 0). It works by iterating ("marching") over a uniform grid of cubes superimposed over a region of the function. If all 8 vertices of the cube are positive, or all 8 vertices are negative, the cube is entirely above or entirely below the surface and no triangles are emitted. Otherwise, the cube straddles the function and some triangles and vertices are generated. Since each vertex can either be positive or negative, there are technically 28 possible configurations, but many of these are equivalent to one another.

2.1 Complexity

In marching cubes algorithm, the computational complexity that is approximately O(n^3)(is the length of the volume) is acquired. Time complexity is O(n), where n is the total number of processed cells.

2.2 Pseudo code

Input: Volume data V and an iso value h. Output: List of vertices to be rendered, together with their respective normal.

1. for each voxel (cube) v of V do
2. Calculate an index to the cube, comparing the 8 density values of the cube vertices with the iso value h.
3. Using the calculated index verify the edge list from a lookup- table.
4. Using the scalar values in each vertex of the edge, find the surface-edge intersections by linear interpolation.
5. Calculate a unitary normal in each cube vertex by the method of central differences. Interpolate the normal to each triangle vertex.

6. Return the triangle vertices and vertex normal.
7. End for.

2.3 Shortcomings

1. Creates mesh which can be highly over-sampled (requires Decimation) and contain bad triangle shapes (requires Mesh Optimization / Re-Meshing)
2. Linear interpolation of distance values approximates smooth surfaces: Sharp features within cells are lost
3. Aliasing artifacts due to alignment of the voxel grid. Increasing voxel resolution only reduces the size of the artifacts but they never go away

2.4 Strengths

The advantage of this algorithm is that the processing of a cell is independent of the other ones, which allows its parallelization. However, as a disadvantage, it may generate holes in the isosurfaces, due to topological ambiguities of the cases.

3 Multi ISO surface extraction algorithm

An isosurface is a three-dimensional analog of an isoline. It is a surface that represents points of a constant value (e.g. pressure, temperature, velocity, density) within a volume of space; in other words, it is a level set of a continuous function whose domain is 3D-space. Isosurface is sometimes used more generically related to domains of more than 3 dimensions.

3.1 Complexity

We develop an isosurface extraction algorithm with a worst-case complexity of O($\sqrt{n+k}$)for the search phase, where n is the size of the dataset and k is the number of cells in the isosurface. The memory requirement is kept at O(n), while the preprocessing step is O(n log n).

3.2 Pseudo code

split(M, T, T_0) - M is the mesh, T is the triangle being split, and T 0 is the triangle that initiated the split.

1. split(M,T, T_0) : T_1 and T_2 are set to the two triangles obtained when T is split
2. M $\leftarrow$ M T_1, T_2 T
3. Update edges in the dual graph
4. if ($T_0$6 =)(T_0 does not share a base edge with T) then
5. set T_0 to whichever of T_1 or T_2 shares an edge with T_0
6. split(M, T_0, T)
7. end if
8. insert(, T_1, T_2)

3.3 Shortcomings

Its computational load is high, and each time a new threshold value is selected, the generation of the new surface may cause delays. The number of triangles produced by the method in a typical set of volume image data is very large, typically on the order of tens of thousands. Thus, displaying them all can be an intensive graphic task. Adaptive surface

extraction techniques were developed to address this problem using an approach that merges coplanar triangles to be represented by a larger polygon. This approach can substantially improve the performance because the number of vertices that need to be processed in the transformation pipeline is significantly reduced. Reduction can be obtained with Delaunay triangulation (also known as thin-plate techniques), where coalescing can be extended to include triangles that are approximately coplanar, within a given tolerance. This can reduce the triangle population at the expense of a negligible drop in quality.

3.4 Strengths

The advantage of this method is that it produces a fairly detailed surface representation for the objects of interest, particularly when the objects are easily separable by their signal distribution in the data. The advantage of this method is that it generates generally less triangles than Marching Tetrahedra method for time and memory comparisons. The advantage of this approach is that it generates final image principally without "holes" that might happen in MC algorithms. 6 tetrahedra scheme gives the advantage of common faces for tetrahedra of neighbour voxels. This is very important feature for iso-surface generation.

4 Bidirectional Chamfer Distance Algorithm

The Chamfer System offers a high-performance solution to shape-based object detection. It covers the detection of arbitrary-shaped objects, whether parametrized (e.g. rectangles and ellipses) or not (e.g. pedestrian outlines). Because the system learns shape distributions of target objects from examples, it is flexible and easily adaptable, without reprogramming. Its pixel-based correlation approach eliminates the need for error-prone contour segmentation. A major advantage is the hierarchical approach of the system (in both shape and transformation space); it results in substantial efficiency gains compared to an equivalent brute-force template matching method. Speed-ups of several orders of magnitude have been measured. The Chamfer System has been employed in a wide range of applications, from object detection on-board smart vehicles, to inspection tasks in industrial vision, and target recognition for the military environment. The Chamfer System is MMX-enabled and runs under Windows/Linux on a standard PC. Depending on the specific application, it runs in real-time or at speeds close to real-time.

4.1 Complexity

In this work, we use Euclidean distances between points for creating our encoding, but other metrics could be used in principle. Since we are working with 3D point clouds (which corresponds to having a small value for d), the nearest neighbor search can be made efficient by using data structures like ball trees . Asymptotically, O(n log n) operations are needed for constructing a ball tree from the point cloud Xi and O(k log n) operations are needed to run nearest neighbor queries for k basis points. This leads to an overall encoding complexity of O(n log n + k log n) per point cloud. The kNN search step can be also efficiently implemented as part of an end-to-end deep learning pipeline. Practically, we benchmark our encoding scheme for different values of n and k and show real-time encoding performance for values interesting for current realworld applications

4.2 Pseudo code

1. Ino= Number of input layers
2. Hno= Number of hidden layers

3. Ono= Number of output layers
4. S= number of data set instances Forward pass
5. for i=1 to Hno
6. for j=1 to S calculating the forward pass for the forward hidden layer's activation function ht using eq.
7. end for
8. for j=S to 1
9. calculating the backward pass for the backward hidden layer's activation function ht using eq.
10. end for
11. end for
12. for i=1 to Ono
13. calculating the forward pass for the output layer using the previous stored activations using eq. (3)
14. end for Backward pass
15. for i= Ono to 1
16. calculating the backward pass for the output layer using the previous stored activations using eq. (3)
17. end for
18. for i=1 to Hno
19. for j=1 to S calculating the backward pass for the forward hidden layer's activation function ht using eq. (5)
20. end for
21. for j=S to 1
22. calculating the forward pass for the backward hidden layer's activation function h; using eq. (4)
23. end for
24. end for

4.3 Shortcomings

Similar to pixels for 2D images, occupancy grid is a natural way of encoding 3D information. Numerous deep models were proposed that work with occupancy grids as inputs . However, the main disadvantage of this encoding is their cubic complexity. This results in a high amount of data needed to accurately represent the surface. Even relatively large grids by our current memory standards (128^3 , 256^3) are not sufficient for an accurate representation of high frequency surfaces like human bodies. At the same time, this type of voxelization results in very sparse volumes when used to represent 3D surfaces: most of the volume measurements are zeros. This makes this representation an inefficient surface descriptor in multiple ways. A number of methods was proposed to overcome this problem . However, the problem of representing high frequency details remains, together with a large memory footprint and low computational efficiency for running convolutions.

4.4 Strengths

We borrow several ideas from these works, such as using kNN-methods for searching efficiently through local neighborhoods or achieving order invariance through the use of pooling operations over computed distances to basis points. However, we believe that the proposed encoding and model architectures offer two main advantages over existing point cloud networks: (a) higher computational efficiency and (b) conceptually simpler, easy-to-implement algorithms that do not rely on a specific network architecture or require custom neural network layers. Compared to other encodings of point clouds, the proposed representation also has an advantage in being more efficient with the number of values needed to preserve high frequency information of surfaces.

While showing competitive performance to the state-of-theart methods on the FAUST dataset, the main advantage of our method is the ability to produce an aligned high resolution mesh from a noisy scan in a single feed-forward pass. This can be executed in real time even on a non-GPU laptop computer, requiring no additional post-processing steps. We make our code for both presented tasks available, as well as a library for usage in other projects.

5 Conclusion

Thus,the works inspires methods in computer graphics and associated industries such as gaming and animation, to generate meshes that require significantly less human intervention for rendering and simulation by using some hihgly welldefined algorithms. The proposed NMF method addresses an important need that has not been adequately studied in a vast literature on 3D mesh generation. While NMF is a first step in addressing that need, it tends to produce meshes that are over-smooth (also reflected in other methods sometimes obtaining greater geometric accuracy) through complex algorithms, which might have a positive impact in applications such as manufacturing.