November 2019

Approximating 3D poly mesh virtual objects with primitive shape colliders

Chloe Snyder

Alper Gungormusler

Andrew Allen

Aleksandr Ayvazov

Dobromir Yordanov

Approximating 3D poly mesh virtual objects with primitive shape colliders

ABSTRACT

A seamless immersive experience requires appropriate physical interactions between 3D objects in the scene being viewed. Systems typically handle such interactions by detecting and resolving collisions between 3D objects in the scene. However, collision detection between two high poly meshes of the two colliding 3D objects has the computational order of $O(N^2)$ where N is the triangle count for the respective meshes, thus making the computations time-consuming which prohibits real-time use. Current techniques include manual creation of simpler collider shapes such as spheres and boxes to approximate the geometry. This disclosure describes techniques that automatically approximate the poly mesh of any complex 3D virtual object with a set of colliders that are primitive shapes such as spheres, cubes, capsules, etc. The application of the techniques to automatically generate colliders enable leveraging mechanisms of collider-based collision detection which can be several orders of magnitude faster than collision detection between poly meshes. As a result, the described techniques can be used for offline pre-processing that provides output that enables real-time collision detection, without human intervention, with reduced consumption of computational resources, thus improving the user experience and overall performance of AR/VR/MR/ER software and/or hardware.

KEYWORDS

- Augmented Reality (AR)
- Virtual Reality (VR)
- Mixed Reality (MR)
- Extended Reality (ER)
- Virtual object

- Primitive shape
- Poly mesh

BACKGROUND

Technologies such as Augmented Reality (AR), Virtual Reality (VR), Mixed Reality (MR), or Extended Reality (ER) are deployed to create immersive user experiences (UX) that involve 3D objects in the user's virtual or augmented environment or game. A seamless immersive experience requires appropriate physical interactions between 3D objects in the scene being viewed. To this end, systems such as game engines typically handle such interactions by detecting and resolving collisions between 3D objects in the scene. However, collision detection between high poly meshes of two colliding 3D objects has the computational order of $O(N^2)$ where N is the triangle count for the respective meshes. It is not uncommon for object models to contain thousands of triangles, thus making the computations time-consuming which prohibits real-time use.

The computational complexity of the collision detection operation can be reduced via various techniques that simplify high poly meshes for collision testing. The most computationally efficient mechanism of reducing intersection costs involves replacing the meshes with a small number of colliders represented as simple primitive shapes, such as spheres, cubes, cylinders, etc. Approximating a mesh with a small number of such primitive shape colliders drastically reduces the computational complexity of collision detection since collection detection and resolution of a primitive shape is computationally fast. Currently, designers of 3D objects must generate the primitive shape mesh approximations manually, thus making the process tedious and time consuming.

Automating the process of breaking up a complex poly mesh into simpler primitive shapes is typically achieved via two main techniques: classical and machine learning. Classical approaches include shape decomposition methods such as breaking up a large complicated object into smaller objects called Generalized Cylinders (GC). Similarly, mesh segmentation techniques are applied to break up a mesh into simpler regions. However, collision computations using the GCs or simpler regions can still be computationally expensive as these are not primitive shapes. As a result, the GCs or simpler regions would further need to be approximated with simple colliders to be viable in high performance, real-time systems, which drastically reduces the advantages of these approaches.

In contrast, machine learning approaches use relevant data to accomplish the task of fitting a mesh with primitive shapes. For instance, such algorithms may rely on statistical analyses that indicate the likelihood that a given labeled object resembles another labeled object. Solutions that employ supervised machine learning (e.g., [5]) to recognize the primitive shapes that compose a complex object model require a database of pre-labeled 3D assets, e.g., an object labeled as ‘chair’ with appropriate sub-labels for its corresponding parts, such as ‘leg,’ ‘seat,’ ‘seatback,’ etc. In contrast, solutions that employ unsupervised machine learning (e.g., [1], [4]) do not require such a database of pre-labeled 3D assets. However, unsupervised approaches require large databases with labeled object categories such as ‘cat,’ ‘chair,’ ‘bed,’ etc. Moreover, machine learning algorithms may not work well with assets that are assigned open-ended category labels. For example, a mesh labeled ‘robot,’ could resemble a human or be shaped like a vacuum cleaner. The set of approximated primitives for these two types of ‘robot’ forms would be drastically different. Further, neither supervised nor unsupervised approaches can be applied on objects that cannot be labeled, such as an abstract work of art.

Additionally, machine learning algorithms have been applied to identify and fit primitive shapes to noisy data of 3D point clouds. Many of these approaches (e.g., [2], [3]) rely on variations of the RANSAC algorithm. Alternatively, convolutional neural networks (CNNs) can be used to segment the point cloud, subsequently using the segment boundaries to define probable primitive shapes that fit. These algorithms are used to generate mesh data from the point cloud by smoothing out noise and filling in the regions omitted by the point cloud. As such, these approaches are not suitable for the task of meaningful simplification of a given mesh with primitive shapes.

DESCRIPTION

This disclosure describes techniques that approximate the poly mesh of any given complex 3D virtual object with a set of colliders that are primitive shapes such as spheres, cubes, capsules, etc. The colliders can then be used in operational implementations of various kinds of AR/VR/MR/ER environments or any 3D context.

The conversion of a 3D object from a poly mesh to an approximate set of primitive-shape colliders is achieved using the following process:

1. Import the complex poly mesh of the 3D object to be approximated.
2. Voxelize the imported mesh to an arbitrary resolution.
3. Group the voxels into a desired number of distinct clusters with a suitable clustering algorithm such as k-means clustering, Gaussian mixture models, hierarchical clustering, density based clustering (e.g., DBSCAN), etc.
4. Compute the origin of each primitive shape that fits each cluster.

5. Fit each cluster with one or more primitive shape in an optimal manner that selects the most appropriate primitive shape and most appropriate transform corresponding to that shape.
6. Export the optimized set of primitive-shape colliders for operational purposes.

The appropriate number of primitive shapes is determined on the basis of relevant heuristics. Optimization of the selection of the most appropriate primitive shapes and corresponding transform in step 5 above is achieved by minimizing the following function:

$$\min_{\vec{l},\vec{\mathbf{M}}} \|\mathcal{T} - \sum_{c=1}^{C} \mathcal{A}_c(l_c, \mathbf{M}_c)\|, \quad \mathcal{A}, \mathcal{T} \in \mathbb{N}^{I \times J \times K}, \quad \mathbf{M} \in \mathbb{R}^{4 \times 4}$$

where:

- *T* and *A* are the target and approximation tensor voxel sets, respectively;
- *I, J,* and *K* are respectively the width, height, and depth of the tensors in voxels;
- *C* is the total number of clusters/primitive shapes;
- *M* is a standard 3D transform matrix; and
- *l* is the type of primitive shape.

The optimal fit contains the sets {*l*} and {*M*} that respectively contain the optimal *l*s and *M*s for each cluster approximation.

A scoring metric (e.g., a voxel-by-voxel scoring metric, or any combination of tensors as outlined above) is used to determine the primitive shape that best fits a cluster of voxels. The metric is applied to find the maximal fit for a given cluster and primitive shape. To that end, a target set of voxels is compared to a set of approximated voxels generated from a candidate primitive shape and transformation. To calculate a simple candidate metric, the candidate voxel

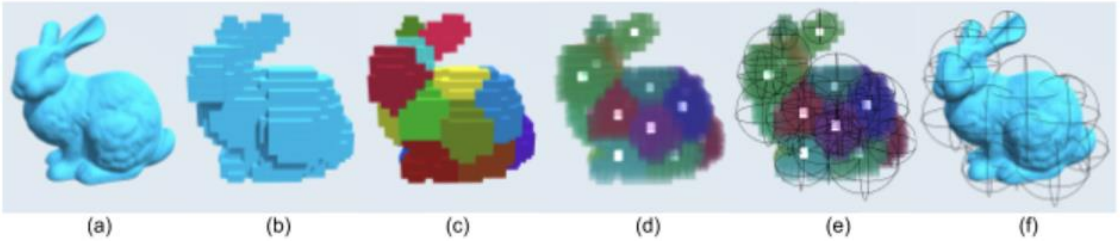
set and a voxel set that gives an approximate representation of a simple collider shape candidate in the arbitrary voxel space, as described above are defined. These two sets can then be evaluated against each other via a simple diffing algorithm with scoring reward/penalties.

In an example solution, the number of points to add or subtract from the metric score can be determined by a suitable heuristic function. For instance, as shown in Table 1 below, voxel matches between the target and in approximation are rewarded (boxes with +) while mismatches are penalized (boxes with -). Thus, empty voxels in the target that correspond to empty voxels in the approximation are rewarded. Penalties are assigned an empty voxel in the target matching filled voxels in the approximation and vice versa.

	Air in Approximation	Current Cluster Filled in Approximation	Other Cluster Filled in Approximation
Air in Target	+	-	-
Current Cluster Filled in Target	-	+	+
Other Cluster Filled in Target	-	+	+

Table 1: Rewards and penalties based on matches and mismatches between voxels in the target and approximation

The primitive shape and transformation that achieves the highest score is selected as the optimal fit for that cluster and added to the set of candidate colliders.



(a) The Stanford Bunny; (b) voxelized with 2.5% relative resolution; (c, d) clustered into 18 regions; (e, f) producing 18 sphere colliders

Fig. 1: Approximating a 3D poly mesh with colliders of primitive shapes

Fig. 1 shows the application of the steps described above to create a primitive-shape approximation of the 3D poly mesh of the Stanford Bunny object (Fig. 1a). The object is voxelized with 2.5% relative resolution (Fig. 1b) and subsequently clustered into 18 distinct regions (Figs. 1 c and d). Each of the 18 regions is then approximated with a corresponding sphere (Figs. 1 e and f), which is a primitive shape.

The application of the techniques described herein can significantly reduce the computational complexity of collision generation at runtime, often by several orders of magnitude. The reduction in computation complexity requires a tradeoff with accuracy. However, the corresponding reduction in accuracy required to reduce computational complexity is relatively small and typically tolerable. Further, exact computations can be performed within a local region whenever higher accuracy is necessary. As a result, the described techniques can be used for offline pre-processing that provides output that enables real-time collision detection, without human intervention, with reduced consumption of computational resources, thus improving the user experience and overall performance of AR/VR/MR/ER software and/or hardware.

The techniques can be implemented within any software and/or hardware system that involves the use of AR/VR/MR/ER or other 3D graphics to deliver immersive real-time 3D experiences. Such systems include software such as AR developer platforms, authoring tools, game engines, maps, etc. and hardware such as VR goggles or headsets. By automating the creation of collider meshes, the proposed techniques minimize the need for designers and developers to create manual approximations, thus saving them time and effort. Given the right representation, the techniques described herein can be used to generate a first-pass collider set that can then be adjusted manually to provide a more streamlined hybrid algorithmic-and-human

approach. The techniques can be integrated within a system and/or packaged in the form of developer tools such as an Application Programming Interface (API). The techniques can also be used in other contexts, e.g., traditional video games, movie visual effects, 3D simulations, etc.

CONCLUSION

This disclosure describes techniques that automatically approximate the poly mesh of any complex 3D virtual object with a set of colliders that are primitive shapes such as spheres, cubes, capsules, etc. The application of the techniques to automatically generate colliders enable leveraging mechanisms of collider-based collision detection which can be several orders of magnitude faster than collision detection between poly meshes. As a result, the described techniques can be used for offline pre-processing that provides output that enables real-time collision detection, without human intervention, with reduced consumption of computational resources, thus improving the user experience and overall performance of AR/VR/MR/ER software and/or hardware.

REFERENCES

1. Huang, Qixing, Vladlen Koltun, and Leonidas Guibas. "Joint shape segmentation with linear programming." In ACM Transactions on Graphics (TOG), vol. 30, no. 6, p. 125. ACM, 2011.
2. Kang, Zhizhong, and Zhen Li. "Primitive fitting based on the efficient multiBaySAC algorithm." PloS one 10, no. 3 (2015): e0117341.
3. Li, Yangyan, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. "Globfit: Consistently fitting primitives by discovering global relations." ACM Transactions on Graphics (TOG) 30, no. 4 (2011): 52.

4. Tulsiani, Shubham, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. "Learning Shape Abstractions by Assembling Volumetric Primitives." In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1466-1474. IEEE, 2017.
5. Zou, Chuhang, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. "3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks." In 2017 IEEE International Conference on Computer Vision (ICCV), pp. 900-909. IEEE, 2017.
6. Zhou, Yang, Kangxue Yin, Hui Huang, Hao Zhang, Minglun Gong, and Daniel Cohen-Or. "Generalized cylinder decomposition." ACM Transactions on Graphics (TOG) 34, no. 6 (2015): 171.