Adaptive on-the-fly molecular ribbons generation

P. Hermosilla[1,2], V. Guallar[2], A. Vinacua[1] and P.P. Vázquez[1]

[1]Universitat Politècnica de Catalunya
[2]Barcelona Supercomputing Center

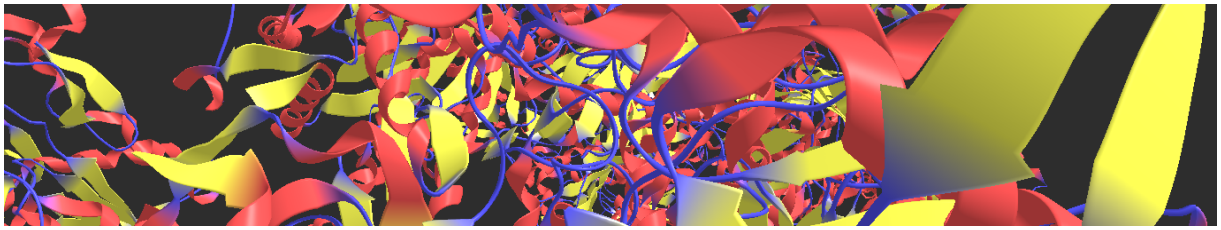

Figure 1: *Secondary structure visualization. The geometry rendered in yellow represents the residues that belong to a β-sheet structure, the one rendered in red represents the residues that belong to an α-helix structure and the geometry rendered in blue represents the residues that do not belong to any type of secondary structure.*

Abstract

Molecular dynamics simulations are of key importance in the drug design field. One common used representation to visualize these simulations is the Ribbons representation, which gives to the expert a good overview of the conformation of the molecule. Although there are several techniques to visualize this representation, all of them have limitations –in terms of space or calculation time, making them not suitable to a real-time interaction with simulation software. In this paper we present a novel adaptive algorithm that generates in real time, using the tessellation shader, the ribbon representation of a molecule without any pre-process.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

The Ribbons model is a popular representation of molecules where the secondary structures are rendered as a set of simple geometries that make them easy to recognize by the experts –Figure 1. Traditionally, these geometries were generated on a preprocess, as in the algorithm proposed by Halm et al. [HOF04] where they generate, in the CPU, only the geometry necessary for the current point of view. Recently, some approaches use the GPU to generate and render them on the fly. For example Krone et al. [KBE08] use the Geometry shader to generate the ribbon's geometry in real-time with a small pre-process of the molecule and Wahle et al. [WB11] propose a similar approach but using only the vertex shader state. Unfortunately, those methods do not scale well because the amount of geometry generated grows linearly with the number of residues of the molecule. Other approaches represent the secondary structures by impostors – [BDST04, BSN12]. Despite the high frame rates obtained with these methods the visual quality is not as good as that of the geometry-based methods.

In this paper we present a new algorithm that uses the GPU tessellation stage to generate ribbons on demand, with a precision that depends on their relative position with respect to the observer. It is based on the algorithm proposed by Carson [Car91], where the ribbons are built using B-Splines whose control points are defined by the C_α-atoms of each residue. Our algorithm determines the type of sec-

ondary structure that each segment belongs to and generates, on the fly, only the necessary geometry to represent it.

2. Algorithm and Results

Algorithm We are specially interested in the visualization of molecules during simulations, in which the positions of the atoms change at every frame, making pre-processing prohibitive, and frame-rate critical. Ribbons are rendered as a series of ribbon helix, flat arrows and tubes that follow the backbone of the biomolecule, with different color for each type of secondary structure –Figure 1. In our algorithm these geometries are represented using a set of patches of different size.

As in [WB11] and [KBE08], we use a B-Spline to have a smooth representation of the backbone. In particular, we use the algorithm proposed by Carson [Car91], where the control points of the B-Spline are defined by the position of the C_α-atoms of each residue.

Our algorithm determines the tessellation level required for each B-Spline segment using the distance from the control points to the camera and subdivides them accordingly. The result of this process is a smooth representation of the backbone that only generates the needed segments to represent the curve for the current point of view. Following that idea, each processed segment is extruded in order to generate the secondary structure geometry and this geometry is then subdivided according to the camera and extrusion distances. This extrusion process is guided by two vectors that determine the displacement directions, d and n – Figure 2.

Applying different scale values to d and n for each type of secondary structure, the algorithm generates a continuous geometry along the backbone that represents the secondary structure conformation of the molecule.

We have implemented this algorithm fully in the GPU taking advantage of the tessellation unit, so there is no need to pre-process the molecule and the stored data is minimized.

Implementation When the first step of the simulation is received we create a vertex and index buffer. In the vertex buffer we store the control points with the following information: Position of the C_α-atom of the residue, the direction of the d vector –some of these vectors have to be flipped to avoid angles larger than 90 degrees between consecutive directions– and an int that encodes the type of secondary structure that the residue belongs to. In the index buffer, we store four indexes for each B-Spline segment that define the four control points used to evaluate them –Figure 2. Note that with this configuration only the vertex buffer has to be recomputed for each new step calculated by the simulation.

This information is sent through the graphics pipeline where each segment is interpreted as a patch with four vertices. The first step of the pipeline, the vertex shader, determines for each vertex which color is going to be used to

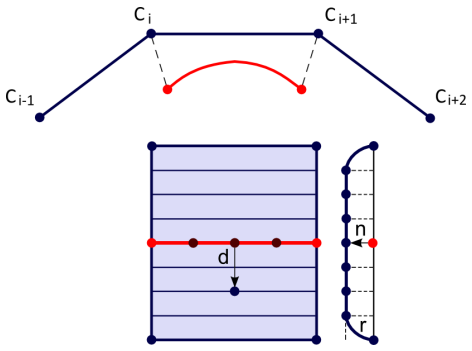

Figure 2: *The B-Spline segment –red color– is subdivided according to the distance to the observer. Then, the subdivided segment is used to create the final geometry. Each segment generates a curved patch using the alignment direction of the secondary structure –d, which, in our algorithm, is defined as the direction between the C_α-atom and the O-atom of each residue [CB86], and a vector perpendicular to d and the segment direction –n.*

shade this part of the backbone. In the tessellation control shader, we calculate the tessellation level of the geometry and B-Spline and the n vectors of control points . Next, the tessellation evaluation shader, using the vertex coordinates generated by the tessellation unit, perform the interpolation to determine the final position and color of each new vertex created. The final step of the pipeline, the fragment shader, calculates the normal of each pixel and shade the geometry with the light information.

This algorithm generates a patch for each segment of the backbone, but from some points of view, it will be necessary to generate the patch that is oriented in the opposite direction too. In order to do so, we increase the index buffer size and duplicate the indexes of each segment in the opposite direction. With this small modification we are able to generate all the patches with the n vectors and segment directions inverted without modifying the algorithm.

Results The algorithm presented in this paper improves the generation of Ribbons on the fly, allowing the interaction in real-time with big molecules. With this technique we achieve framerates of 582-540-463 for far-medium-close views of molecules of 545040 atoms and 71100 residues on a computer with an i7 CPU, 12Gb of RAM, and an nVidia GTX770 graphics card at a resolution of 1376x961. This method only requires storing a few bytes for each B-Spline segment, making it suitable for the real-time visualization of molecular dynamics simulations.

Acknowledgements This work has been supported by the projects TIN2013-47137-C2-1-P and TIN2014-52211-C2-1-R of the Spanish Ministerio de Economía y Competitividad, and the project 2014-SGR 146 from the Catalan Government.

References

[BDST04] BAJAJ C., DJEU P., SIDDAVANAHALLI V., THANE A.: Texmol: interactive visual exploration of large flexible multi-component molecular complexes. In *Visualization, 2004. IEEE* (Oct 2004), pp. 243–250. doi:10.1109/VISUAL.2004.103. 1

[BSN12] BAGUR P. D., SHIVASHANKAR N., NATARAJAN V.: Improved quadric surface impostors for large bio-molecular visualization. In *ICVGIP* (2012), Triggs B., Bala K., Chandran S., (Eds.), ACM, p. 33. 1

[Car91] CARSON M.: Ribbons 2.0. *Journal of Applied Crystallography 24*, 5 (1991), 958–961. 1, 2

[CB86] CARSON M., BUGG C. E.: Algorithm for ribbon models of proteins. *Journal of Molecular Graphics 4*, 2 (1986), 121–122. 2

[HOF04] HALM A., OFFEN L., FELLNER D.: Visualization of complex molecular ribbon structures at interactive rates. In *Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on* (July 2004), pp. 737–744. doi:10.1109/IV.2004.1320224. 1

[KBE08] KRONE M., BIDMON K., ERTL T.: GPU-based Visualisation of Protein Secondary Structure. In *Theory and Practice of Computer Graphics* (2008), Lim I. S., Tang W., (Eds.), The Eurographics Association. doi:10.2312/LocalChapterEvents/TPCG/TPCG08/115-122. 1, 2

[WB11] WAHLE M., BIRMANNS S.: Gpu-accelerated visualization of protein dynamics in ribbon mode. In *IS&T/SPIE Electronic Imaging* (2011), International Society for Optics and Photonics, pp. 786805–786805. 1, 2