EG UK Theory and Practice of Computer Graphics (2008)
Ik Soo Lim, Wen Tang (Editors)

GPU-based Visualisation of Protein Secondary Structure

Michael Krone[1] and Katrin Bidmon[1] and Thomas Ertl[1]

[1]Visualisation Research Center (VISUS), Universität Stuttgart, Germany

Abstract

The increasing number of protein 3D structure information available makes the high-quality visualisation of this information play a more and more important role. Beside element-based representations the secondary-structure-based representation – also called cartoon representation – refers to a higher level of abstraction, representing the protein structure as tubes and ribbons. We present a method for this cartoon representation of proteins using the ability of modern graphics hardware's geometry shaders and thus reducing the amount of data to be transferred from the CPU to the graphics card. The resulting minimisation of storage needed is of particular importance when dealing with huge datasets. High-quality images at interactive frame rates can thus be achieved.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Line and Curve Generation I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling I.6.6 [Simulation and Modeling]: Simulation Output Analysis

1. Introduction

Proteins are the foundation of nearly all procedures in living organisms. They are not only forming the principal component in all cells but take part in every biochemical process. Hence, there is a wide range of research areas and applications where high quality protein representations are needed. The information about the basic configuration of each protein is obtained by X-ray crystallography at high resolution. Based on this information several representations, emphasising different characteristics each, established. A detailed description of the protein structure is given in the following Section 1.1.

The most common protein representations can be split into atom- or primary-structure-based representations such as *ball-and-stick*, *stick*, *space-fill* or surface representations and the so-called *cartoon* representation, based on the secondary structure of the protein and visualising a higher level of abstraction of the molecule. Figure 1 shows a protein in stick and cartoon representation.

The potentially huge size of these proteins, often visualised together with other structures such as surrounding solvent molecules, depending on the application field, raises a challenge to provide a visualisation at interactive frame rates. Therefore, we decided to fathom the potential of modern graphics hardware's geometry shaders for this aim and thus to reduce the data that has to be transferred to the graphics card for rendering.

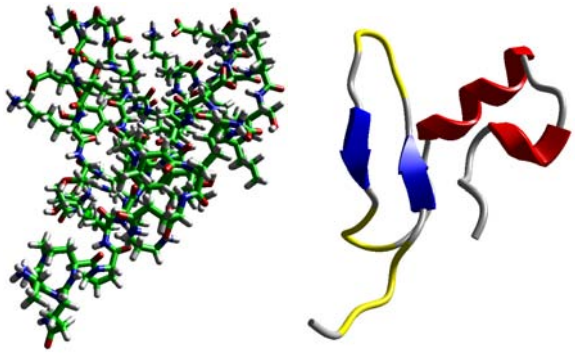


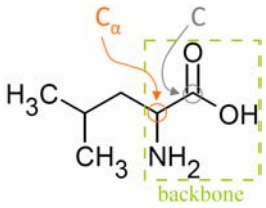
Figure 1: *Exemplary protein representations of the same protein: stick (left) and cartoon (right).*

This paper is structured as follows: Section 1.1 provides an introduction to the molecular biological background of proteins and the visualised protein structure followed by some details about the underlying data in Section 1.2. An overview of related work is given in Section 2. The basic components of the cartoon representation are introduced in Section 3, followed by their geometrical construction il-

lustrated in Section 4 and implementation details in Sections 4.1, 4.2 and 4.3. Section 5 is dedicated to exemplify the results achieved and to state some performance information. Section 6 concludes this paper.

1.1. Molecular Biological Background

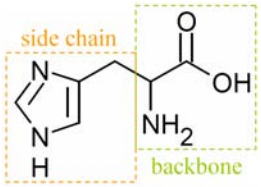
Proteins are linear long-chained macromolecules consisting of standard amino acids. The length of the chain varies from only very few to several thousand amino acids. The average chain length however is 300 amino acids. There are twenty standard amino acids which consist of the elements carbon, nitrate, oxygen, hydrogen and, in some cases, sulphur. All amino acids have an identical part, called *backbone*, which consists of the amino and the carboxyl functional group, which are attached to another carbon, the so-called α-*carbon (Cα)* which is depicted in Figure 2.



Leucine (Leu)

Figure 2: *Chemical structure of the amino acid leucine with the carbon Cα marked orange and the carbon C of the carboxyl group marked grey.*

The *side chain* of the amino acid is also attached to the Cα-atom (confer Figure 3). This side chain is unique to every amino acid. Each amino acid's amino group is connected to the carboxyl group of its successor. Thus, the chain or strand of the protein is formed. Arborisation never occurs with proteins.



Histidine (His)

Figure 3: *Chemical structure of the amino acid histidine. The backbone with the carboxyl group (COOH) and the amino group (NH_2) is framed green, the side chain (imidazole) is framed orange.*

The sequence of amino acids, which is unique to each protein, is called the protein's *primary structure*. The side chains of the amino acids are responsible for the proteins' biochemical characteristics such as hydrophobicity or hydrophilicity. Furthermore, the amino acids can form bonds between each other like hydrogen bonds, salt bridges and disulphide bonds. That way the strand folds in a specific manner according to the energetic minimum. The folding is highly stabilised by these linkages [PC51]. This conformation is called the protein's *secondary structure*. Folding is essential for the protein's functionality: many diseases like Alzheimer's or BSE are caused by misfolded proteins.

There are several folding patterns, called *secondary structure elements*. The most common secondary structure elements are α-*helices* where the strand forms a right-handed helix as shown in Figure 5a, β-*sheets* which are formed by interconnected parallel strands as shown in Figure 5b and *turns* where the strand undergoes a 180° bend. Unspecific regions in the strand are called *random coil*. The three-dimensional alignment of the secondary structure elements is the *tertiary structure*. Multiple folded proteins can be associated by the same bonds as the secondary structure, forming a functional complex which is termed *quaternary structure*. To simplicity matters, tertiary and quaternary structure are often also referred to as secondary structure.

1.2. Data

The data used for the visualisation examples in this paper originate from the RCSB Protein Data Bank [BWF*00] which is part of the wwPDB [wwP08]. The data was obtained in the PDB file format [wwP07], a simple and easy to parse ASCII file format. In a PDB file only the primary structure information e.g. the sequence of the amino acids and the coordinates of their atoms is mandatory, secondary structure information is optional.

Therefore, the software STRIDE by Argos and Frishman [FA95] is used to derive secondary structure information from the atomic coordinates given in a PDB file. It uses the same method as DSSP by Kabsch and Sanders [KS83] but utilises not only hydrogen bond energy but also main chain dihedral angles to obtain the secondary structure.

2. Related Work

The visualisation of macromolecular structures is a busy research area. Starting from early works about the definition of molecular surfaces [Ric77] and their efficient calculation and representation [Con83, VBW94] nowadays work mainly deals with the interactive high-quality representation of the primary structure of these large molecules – often exploiting modern GPU's programmability. Just to mention a few of them, Lampe et al. [LVRH07] recently presented a new approach for visualising proteins using a point-based method. For a better understanding of the three-dimensional structure, Tarini et al. [TCM06] presented a GPU-accelerated method for applying ambient occlusion to the real-time visualisation of molecules in space-fill mode. Other approaches are more focused on visualising the conformational changes of the molecules (e.g. [SEBH02]).

Additionally, there are various tools for the visualisation of protein-solvent trajectories from MD simulations such as VMD [HDS96], PyMOL [DeL02], MoilView [SEZ95], MD Display [CSL96] and Chimera [PGH*04]. Most of them are being developed over a long period of time and hence offer a wide range of functionality for analysing the protein. Their visualisation techniques are often outdated and suffer from performance problems for large datasets. On the other hand there is a growing number of GPU-optimised viewers like the BioBrowser [HOF05] and TexMol [BDST04] which feature good performance and rendering quality, but the latter, for example, gives a very uncommon cartoon representation substituting the typically used ribbon representing the sheet and tubes representing the coil by spheres.

The secondary structure was primarily predicted and categorised by Pauling and Corey [PC51]. Levitt and Greer later presented a method of automatic assignments of secondary structure elements in [LG77]. Richardson [Ric81] structured and illustrated the prior found characteristics as ribbons and sheets, being common practice today. A good overview of the theory of the secondary structure of proteins is given in [KS83].

3. Structure of the Cartoon Representation

The cartoon representation visualises the secondary structure of a protein. There are special graphical representations for the different kinds of secondary structure elements. As can be seen in Figure 4 the random coil and turns are visualised as tubes, the α-helices are depicted using broad ribbons that coil following the turns of the helices and the β-sheets are drawn as flat, band-shaped arrows that point in the direction of the carboxyl group.

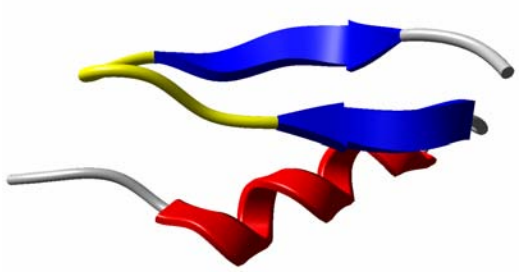
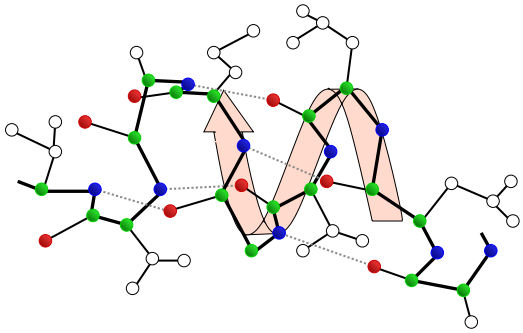


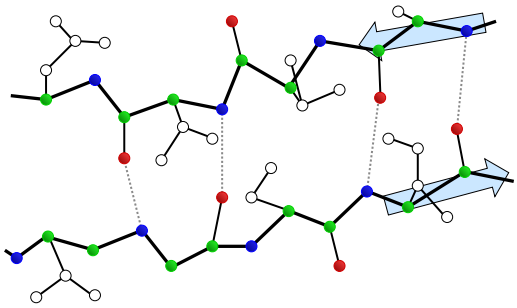
Figure 4: *Protein coloured by secondary structure: the α-helix is depicted red, the β-sheets represented by blue ribbon-shaped arrows, the turn by yellow tubes and the random coil is coloured grey.*

There is no exact definition how the secondary structure elements should be visualised apart from the definitions above. This is the reason why cartoon representations look a little bit different in nearly all available molecular viewers. We have opted for the common representation namely ribbons with rounded edges for helices, ribbons with sharp edges and arrowheads for sheets and tubes for random coil and turn.

The characteristics of the secondary structure follows the backbone of the protein, completely disregarding the side chains. Since only the general structure of the protein is of interest the secondary structure does not need to interpolate the exact positions of the backbone atoms. A smooth curve that merely approximates the gradient of the backbone provides the basis for all secondary structure elements.



(a) Schematic view of an α-helix.



(b) Schematic view of a β-sheet.

Figure 5: *Schematic views of secondary structure elements. Carbon atoms are coloured green, oxygen atoms red and nitrate blue. The backbone bonds are illustrated using thick black lines. Side chains are drawn as outline. Dotted grey lines illustrate hydrogen bonds.*

An important feature of the ribbons used for visualising helices and sheets is their alignment. The ribbon always lies in one plane with the hydrogen bonds that connect and stabilise the structure (confer [Ric81]). These hydrogen bonds always form between the oxygen atom of the backbone of one amino acid and the nitrate atom of another. Schematic views of an α-helix and a β-sheet are shown in Figure 5. The position and alignment of the secondary structure ribbons in relation to the position of the atoms are shown as broad red (helix) respectively blue (sheet) arrows. Thereby the ribbons of the parallel strands of a β-sheet are aligned to

the same plane, optically forming the eponymous sheet. The ribbon of the helix seems to wind flatly around an invisible tube which fills the inside of the helix.

Because the cartoon representation only visualises the structure of the protein most customary colouring methods for molecules such as colouring by element or colouring all atoms of one amino acid in the same colour are not suitable. The most common colouration for cartoon is to assign a certain colour to each type of secondary structure element. Another alternative is rainbow colouring whereat the structure of the protein is coloured such that it results in a colour gradient along the chain. The rainbow colour mode does not describe any biochemical attribute but enables a human observer to perceive the distance between certain regions in the chain easily.

4. Cartoon Rendering

Cubic B-splines were chosen for the generation of the curve the secondary structure is based upon. By using cubic B-splines a satisfying compromise between calculation time and output quality can be achieved and a smooth curve is computed that follows the characteristics of the backbone sufficiently accurate. We have used the B-spline algorithm suggested by Carson et al. [Car91]. Control points of the curve sections are the coordinates of the C_α-atoms of four successive amino acids. To create curve sections that join smoothly the last three coordinates of the preceding section plus the coordinates of the subsequent C_α-atom are used. Thus each curve section forms the curve between two C_α-atoms. The number of line segments approximating the curve progression for each section is controlled by parameter.

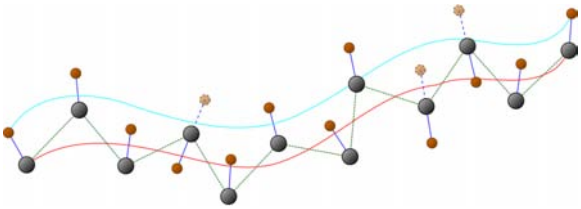


Figure 6: *Schematic drawing of the basic B-spline (red) and the second B-spline used for alignment (turquoise). C_α-atoms are coloured grey, the directions to the O-atoms (brown) are coloured blue. Dashed lines denote flipped directions.*

For the computation of the ribbon alignment not the direction of the hydrogen bond but the direction from the C_α-atom to the O-atom of the backbone is utilised. This direction is virtually equal to the hydrogen bond's direction and can be computed easily from the backbone atoms of one amino acid [CB86]. The major advantage in doing so is that no further information about the secondary structure and its stabilising hydrogen bonds is needed. This direction only apply to the corresponding positions of the C_α-atoms i.e. the start and end point of each curve section. To get the direction for all points of the curve's approximating line segments, a second B-spline with the same number of control points and line segments is computed. Control points of this spline are the coordinates of the C_α-atoms with the normalised directions towards the O-atoms added. Before computing the second B-spline the direction of the O-atom must be determined because it can flip. If the angle between two consecutive directions is wider than 90°, the latter must be flipped, i.e. multiplied with (-1) as depicted in Figure 6.

For each line segment the geometric primitives which form the cartoon representation are computed. According to the schematic view in Figure 7 six vertices $p_0, \ldots, p_5$ are needed for the computation of the geometry, namely the end points of the current line $\overline{p_1p_3}$ and of the preceding $\overline{p_0p_1}$ and the succeeding line segment $\overline{p_3p_5}$. Additionally the end points p_2 and p_4 of the corresponding line segment to the current one in the second B-spline are needed.

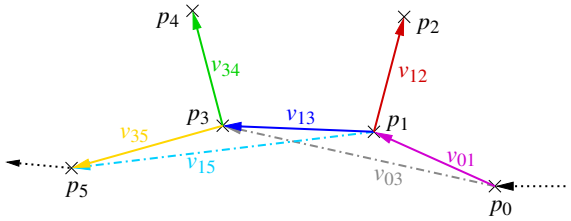


Figure 7: *Vertices $p_1, \ldots, p_5$ and corresponding vectors used to compute the geometry for the current line segment v_{13}*

If the current line belongs to a β-sheet a hexahedron that surrounds the line is created (see Figure 8). The front face is a rectangle that is spanned by the vectors v_{34} (green) and $v_{34} \times v_{15}$ (grey) which are scaled to fit the desired lengths. Analogous the back face is a rectangle that is spanned by the scaled vectors v_{12} (red) and $v_{12} \times v_{03}$ (grey). For the creation of the back face the same vectors as for the creation of the front face of the preceding segment are used. By applying this creation scheme continuous joints without gaps or crossovers are attained. The current line v_{13} (depicted blue in Figure 8) connects the centre of front and back face. For the creation of the arrowhead the width of the front and back faces of all segments of the last curve section of the sheet are scaled according to

$$f_1 = 1 - \frac{j \bmod n_{seg}}{n_{seg} - 1} + k$$

$$f_2 = 1 - \frac{j \bmod n_{seg}}{n_{seg} - 1} + \frac{1}{n_{seg} - 1} + k$$

where f_1 is the scale factor of the front face's width, f_2 of the back face, n_{seg} is the number of line segments for each curve section, $0 \leq j < n_{seg}$ is the index of the current line segment and k is the excess length of the arrowhead.

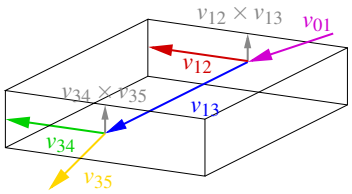


Figure 8: *Geometry for one line segment of a ribbon (β-sheet or α-helix).*

The computation of the ribbon for helices is analogous to the sheet. The only differences are that it has no arrowhead and that its lateral edges are bevelled which results in the desired rounded look. The normals are set in a way that the lighting amplifies this effect. The bevel is $\frac{1}{4}$ of the ribbon's height and $\frac{3}{20}$ of the ribbon's width.

The front and back faces of the tubes that describe the random coil and turns are convex, regular polygons which are created by rotating the vector v_{34} (v_{12} respectively in case of the back face) at uniform angles around vector v_{15} (v_{03} respectively in case of the back face). Figure 9 shows a schematic drawing of the rotation for a hexagon. As with the sheet and helix, the current line segment v_{13} (depicted blue in Figure 9) connects the centres of front and back face which join without gap between two consecutive line segments.

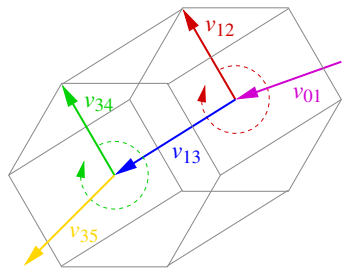


Figure 9: *Geometry for one line segment of a tube (random coil or turn).*

Three methods for rendering the cartoon representation were implemented, a CPU implementation, a plain GPU implementation and a hybrid implementation. The CPU and the hybrid implementation create the same graphical output. The GPU implementation is a testing implementation to analyse the abilities and limitations of current high-end consumer graphics hardware. Therefore, its graphical output is slightly differing from the other two implementations. Further details about the different implementations are discussed in Sections 4.1-4.3.

4.1. CPU Implementation

The CPU implementation was designed as a reference for comparison with the hybrid and GPU implementations. All geometric primitives are computed according to the explanations above and stored in vertex arrays for fast rendering using `glDrawArrays`. Normals and colours are likewise precomputed and stored in arrays. GPU shaders are used only optionally for per-pixel lighting.

4.2. Hybrid Implementation

The hybrid implementation is designed to reduce the computational load of the CPU and the memory requirements significantly. This is an important matter when visualising large time-dependent datasets such as the results of molecular dynamics simulations. Therefore the geometric primitives of the cartoon representation are generated dynamically during each render pass using geometry shader programmes. Only the two splines used as a basis for the computation and the colours are precomputed on the CPU. Thus the number of vertices which have to be stored and passed to the GPU is significantly reduced.

The input primitives for the geometry shader are triangles with adjacency (`GL_TRIANGLES_ADJACENCY_EXT`) introduced with the `GL_EXT_geometry_shader4` extension. This provides the possibility to send the six vertices $p_0, \ldots, p_5$ (confer Figure 7) used to compute a segment to the shader. The geometry shader's output primitives are `GL_TRIANGLE_STRIP`. The computation of the geometry is accomplished analogous to the scheme described in Section 4.

For each type of secondary structure element an individual geometry shader is used. To keep rendering time low the switching of the shaders is minimised by drawing all secondary structure elements of the same type together, so that each shader programme has to be loaded only once to the GPU in each render pass. The time needed for precomputation does not increase noticeably by this sorting, since only the vertices $p_0, \ldots, p_5$ are stored in three different vertex arrays. Since each segment has at most two colours (the colours of the back and the front face) the remaining colour channels are used for passing parameters (such as height and width of the ribbon or the radius of the tube) to the GPU.

Two versions of the shaders are available. They produce the same geometric primitives but one uses per-pixel lighting and the other per-vertex lighting to compare rendering speed. Furthermore, this allows to choose the faster per-vertex lighting instead of the visually superior per-pixel lighting to enable interactive rendering on slower graphics cards.

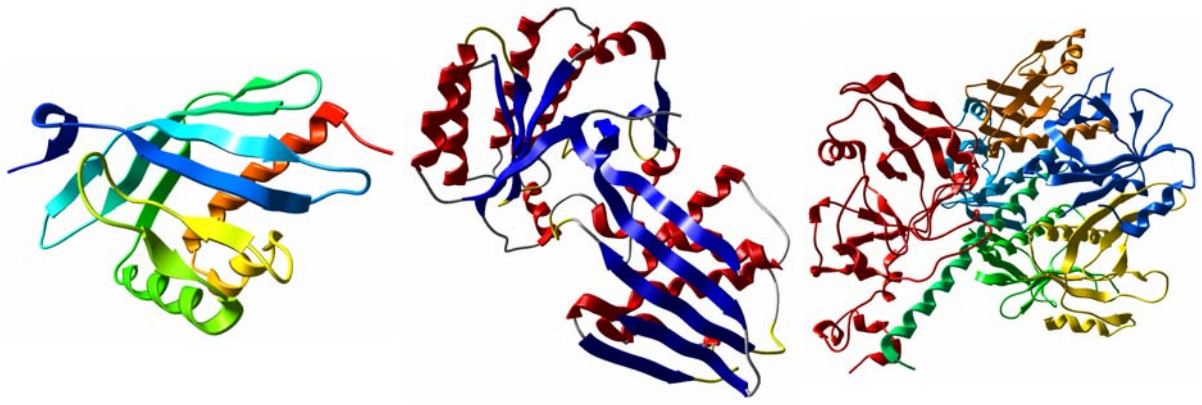


Figure 10: *Results: Rainbow-coloured rendering of isomerase 1OGZ (left), kinase 1VIS coloured by secondary structure (middle), enterotoxin 1TII with each chain coloured apart (right). Different colouring methods are used to emphasise particular attributes of the protein.*

4.3. GPU Implementation

This implementation goes even further than the hybrid implementation in terms of reducing memory costs and precomputation time, since it does not need any kind of precomputation or additional storage. For each curve section, only the positions of the C_α-atoms and the normalised directions towards the O-atoms are passed to the GPU. A geometry shader programme computes the approximating line segments and generates the geometric representation of the secondary structure. As with the hybrid implementation, separate shaders are available for each kind of secondary structure element. The input primitives for the geometry shaders are lines with adjacency (`GL_LINES_ADJACENCY_EXT`) whereby four vertices are sent to the shader. This vertices are the positions of the C_α-atoms of four successive amino acids which form the control points of the B-spline (see Section 4 for details). The normalised directions towards the O-atoms which are used for the computation of the second B-spline and all additional parameters like tube radius, ribbon width and colour are passed to the shader using the primary and secondary colour channels of the vertices. Since no information is stored this parameters are computed for every curve section in each render pass. No sorting of secondary structure elements is applied like in the hybrid implementation, therefore, the appropriate geometry shader programme is newly loaded for each curve section.

The generation of the geometry is widely identical to the hybrid implementation, only the helices are lacking the bevel. The number of line segments for the curve section and the number of vertices for the tubes are parametrised. The approximating line segments of the curve are computed identically to the other implementations. Since only the line segments of the current section are available, the direction v_{35} for the last segment (v_{01} for the first segment respectively) is not available. The direction from the second to the fourth control point (from the first to the third respectively) is used as a substitute.

5. Results and Performance

The visual quality of the cartoon representation is highly dependent on the tessellation. Since all constraints for the appearance are parametrised the user can adjust the graphical output to his needs and the hardware limitations by configuring the number of geometric primitives.

Approximating each curve section with 12 line segments and rendering the tube of random coil and turns with 12-sided front and back faces as shown in Figure 10, a visually satisfying representation, whose quality is close to raycasting, can be achieved.

As evidenced below, cartoon representation can be rendered at interactive frame rates using both CPU and hybrid implementation with per-pixel lighting and high tessellation (confer Table 2). All performance tests were run on an Intel Core2 Duo 6600 with 2 GB DDR2 RAM and an NVidia Geforce 8800 GTX with 768 MB video RAM. The PDB files listed in Table 1 were used as reference datasets for performance measuring. Generally, ATI graphics cards exhibit better performance using geometry shaders with DirectX, but unfortunately still do not support OpenGL. In order to keep

PDB ID	Atoms	Amino acids	File size
1OGZ	944	125	116 KB
1VIS	2,482	324	229 KB
1TII	5,475	738	492 KB
1AF6	10,052	1,263	880 KB
1AON	58,694	6,882	4,700 KB

Table 1: *Datasets used for performance measuring.*

	CPU Implementation per-pixel n=12, i=12	Hybrid Implementation per-pixel n=6, i=6	Hybrid Implementation per-pixel n=12, i=12	Hybrid Implementation per-vertex n=6, i=6	GPU Implementation per-vertex n = 5, i = 6
1OGZ	600 fps	550 fps	240 fps	780 fps	100 fps
1VIS	400 fps	200 fps	90 fps	330 fps	55 fps
1TII	180 fps	150 fps	45 fps	220 fps	28 fps
1AF6	110 fps	100 fps	36 fps	150 fps	16 fps
1AON	19 fps	10 fps	4 fps	15 fps	3 fps

Table 2: *Performance measurements. The number of segments for each curve section is denoted by n, while i is the number of edges of the tube's front and back faces.*

the application platform independent we have decided for an OpenGL implementation and thus are not able to compare the performance on ATI.

The mevalonate kinase 1VIS represents a protein of average size with a chain length of about 300 amino acids while the chaperonin complex 1AON depicted in Figure 11 is one of the largest proteins existing with a chain length of more than 6,500 amino acids.

Table 2 shows the measured frame rates. Although the frame rates for the CPU implementation using traditional precomputed vertex arrays is significantly higher, the hybrid and GPU implementations have several major advantages, especially as a basis for rendering very large, time-dependent datasets.

For example a dataset with the trajectory of a molecular dynamics simulation containing 50.000 time steps can reach a size of about 17 GB. With such large datasets, precomputation time and especially additional storage demands must be minimised to enable interactive rendering. Since the hybrid implementation only pre-computes and stores the vertices and parameters of the two splines instead of all the vertices for the geometry, not only the pre-computation time for each time step is significantly lower but also the memory requirements are reduced at least by half versus the hybrid implementation.

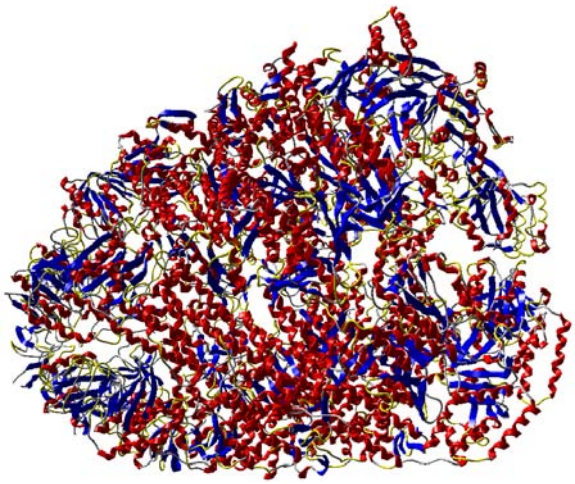


Figure 11: *One of the largest protein structures known: 1AON consisting of about 6,500 amino acids.*

The GPU implementation enhances these advantages by computing also the B-splines on the GPU, therefore, no precomputed information is needed apart from the atom positions loaded from the PDB file. As shown in Table 2 the GPU implementation achieves interactive frame rates for most datasets non-withstanding the increased workload of the GPU.

A major drawback of the GPU implementation is that it was not possible to raise the number of line segments on the available test systems. While the frame rate changed as expected according to the number of segments used for approximation up the values given in Table 2, the frame rate dropped abruptly to less than one fps when further increasing the number of segments. The same behaviour occurred when using per-pixel lighting. Since the number of emitted vertices is far below the maximum number of vertices that the geometry shader can possibly emit, it can be assumed that this is due to internal limitations of the geometry shader. This phenomenon is to be inquired further before advancing the GPU implementation.

Both hybrid and GPU implementation were considerably accelerated by splitting the geometry shader in three separate programmes, one for each secondary structure type instead of using one shader programme with `if/else`-conditions.

6. Conclusions

We presented a method for interactive visualisation of proteins' secondary structure using geometry shaders. Though the CPU implementation still shows better performance, the proposed GPU implementations are a promising basis for supporting time-dependent datasets and yet offering interactive frame rates, reducing or even superseding pre-computations on the CPU. Furthermore, this method requires far less storage, which is of particular importance

when dealing with huge datasets or even protein-solvent-systems.

Nevertheless, there are still phenomena such as the abrupt drop of the frame rate described in Section 5 that need further examination.

7. Acknowledgements

This work is partially funded by the Deutsche Forschungsgemeinschaft (DFG) as part of the Collaborative Research Centre SFB 716.

References

[BDST04] BAJAJ C., DJEU P., SIDDAVANAHALLI V., THANE A.: TexMol: Interactive visual exploration of large flexible multi-component molecular complexes. In *VIS '04: Proceedings of the conference on Visualization '04* (2004), pp. 243–250.

[BWF*00] BERMAN H., WESTBROOK J., FENG Z., GILLILAND G., BHAT T., WEISSIG H., SHINDYALOV I., BOURNE P.: The Protein Data Bank. *Nucleic Acids Research 28* (2000), 235 – 242. http://www.pdb.org/.

[Car91] CARSON M.: Ribbons 2.0. *J.Appl.Cryst. 24* (1991), 958 – 961.

[CB86] CARSON M., BUGG C. E.: Algorithm for Ribbon Models of Proteins. *J.Mol.Graphics 4* (1986), 121 – 122.

[Con83] CONNOLLY M. L.: Analytical molecular surface calculation. *Journal of Applied Crystallography 16*, 5 (1983), 548–558.

[CSL96] CALLAHAN T., SWANSON E., LYBRAND T.: MD Display: An interactive graphics program for visualization of molecular dynamics trajectories. *Journal of Molecular Graphics 14* (1996), 39–41.

[DeL02] DELANO W.: PyMOL: An Open-Source Molecular Graphics Tool. *CCP4 Newsletter On Protein Crystallography 40* (2002). http://pymol.sourceforge.net/.

[FA95] FRISHMAN D., ARGOS P.: *STRIDE: Protein secondary structure assignment from atomic coordinates.* Heidelberg, 1995.

[HDS96] HUMPHREY W., DALKE A., SCHULTEN K.: VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics 14* (1996), 33–38. http://www.ks.uiuc.edu/Research/vmd/.

[HOF05] HALM A., OFFEN L., FELLNER D.: BioBrowser: A Framework for Fast Protein Visualization. In *Proceedings of EUROGRAPHICS - IEEE VGTC Symposium on Visualization 2005* (2005), Brodlie K. W., Duke D. J., Joy K. I., (Eds.).

[KS83] KABSCH W., SANDER C.: Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers 22*, 12 (December 1983), 2577–2637.

[LG77] LEVITT M., GREER J.: Automatic identification of secondary structure in globular proteins. *Journal of Molecular Biology 114* (1977), 181 – 239.

[LVRH07] LAMPE O. D., VIOLA I., REUTER N., HAUSER H.: Two-level approach to efficient visualization of protein dynamics. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (Nov./Dec. 2007), 1616–1623.

[PC51] PAULING L., COREY R. B.: Configurations of polypeptide chains with favored orientations around single bonds: Two new pleated sheets. *Proceedings of the National Academy of Sciences 37* (1951), 729 – 740.

[PGH*04] PETTERSEN E., GODDARD T., HUANG C., COUCH G., GREENBLATT D., MENG E., FERRIN T.: UCSF Chimera - A visualization system for exploratory research and analysis. *Journal of Computational Chemistry 25*, 13 (2004), 1605–1612.

[Ric77] RICHARDS F.: Areas, volumes, packing and protein structure. *Annual Review of Biophysics and Bioengineering 6* (1977).

[Ric81] RICHARDSON J. S.: The anatomy and taxonomy of protein structure. *Advances in protein chemistry 34* (09 1981), 167–339.

[SEBH02] SCHMIDT-EHRENBERG J., BAUM D., HEGE H.-C.: Visualizing dynamic molecular conformations. In *VIS '02: Proceedings of the conference on Visualization'02* (2002), pp. 235–242.

[SEZ95] SIMMERLING C., ELBER R., ZHANG J.: MOILView - A Program for Visualization of Structure and Dynamics of Biomolecules and STO- A Program for Computing Stochastic Paths. *Modelling of Biomolecular Structure and Mechanisms* (1995), 241–265.

[TCM06] TARINI M., CIGNONI P., MONTANI C.: Ambient occlusion and edge cueing for enhancing real time molecular visualization. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1237–1244.

[VBW94] VARSHNEY A., BROOKS, JR. F. P., WRIGHT W. V.: Linearly scalable computation of smooth molecular surfaces. *IEEE Computer Graphics and Applications 14*, 5 (1994), 19–25.

[wwP07] WWPDB: *PDB File Format - Contents Guide Version 3.1.*, 07 2007. http://www.wwpdb.org/docs.html.

[wwP08] wwPDB - World Wide Protein Data Bank. online, March 2008. http://www.wwpdb.org/.