A Client-side Approach towards Platform Independent Molecular Visualization over the World Wide Web

Michael Bender, Hans Hagen, and Axel Seck

bender|hagen|seck@informatik.uni-kl.de
Computer Graphics Group, Department of Computer Science,
University of Kaiserslautern

Abstract. A web-based, entirely platform independent Molecular Visualization System has been developed using state of the art Internet programming techniques. This system offers the visualization of various molecular models, molecular surfaces and molecular properties which can be displayed at the same time. The system itself has been developed using the Java programming language, which allows flexible and platform independent use, perfect integration with the World Wide Web and due to its object-oriented structure easy extension and maintenance. All necessary calculations, e.g. the calculation of a Richards' Contact Surface or an Isosurface, take place on the client's side exploiting the computational power of modern desktop workstations and personal computers.

1 Introduction

From the 70s on, computer molecular representations began to reduce the common limitations of physical molecular models, namely size restriction due to weight and mechanical stability of the models themselves. While the first generations of computers permitted only two-dimensional illustrations for the reason of low computational and graphical power, the development in the last twenty years, led quickly to the kind of scientific visualization we are currently used to.

Commercial and public domain programs offer a variety of molecular representations from simple Stick or Ball-and-Stick Models to advanced surface models like the Van-der-Waals Surface Model or the Richards' Contact Surface Model, allowing easy mapping of molecular properties onto the surfaces. On high-end graphics workstations, e.g. an Onyx2, it is not worth mentioning visualizing hundreds of molecules at the same time, displaying animations in real-time or even computing dynamic interactions with the user in real-time.

Two aspects of applications for Molecular Visualization, not yet considered, are platform independence and – directly related to this – Molecular Visualization over the World Wide Web. These are two main topics this work is focused on. Since highly-optimized computational algorithms and high-speed graphical output depend strongly on the specific underlying system architecture, our approach includes the use of standard methods and libraries to achieve our goal:

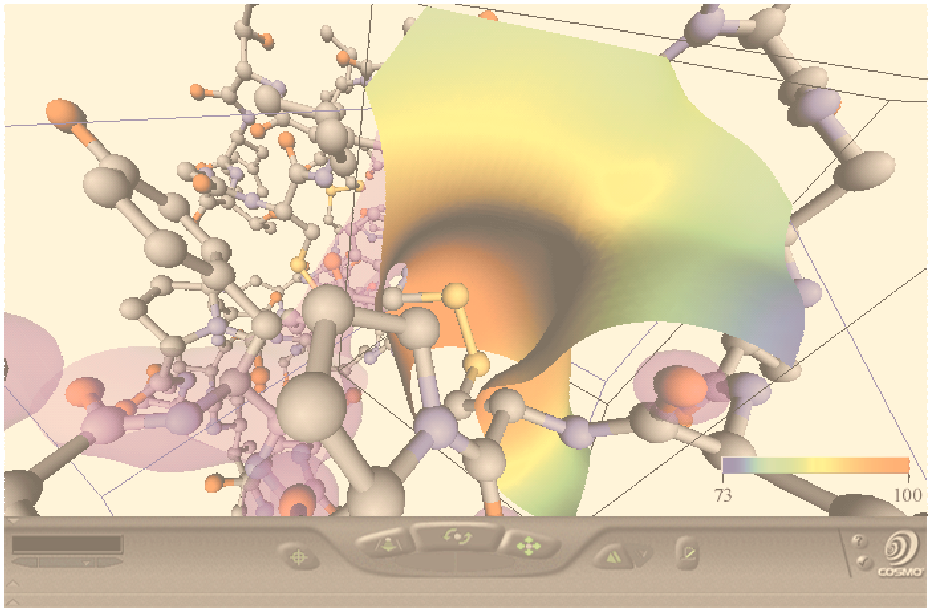

Fig. 1. Plant seed protein *(Ball-and-Stick Model, different Isosurfaces)*

An entirely platform independent web-based tool for displaying molecules and their chemical and physical properties. The molecule data as well as the application itself, which is completely written in Java, are loaded directly by a client from the World Wide Web via an Internet Browser and the application is then executed by the client doing all necessary calculations for visualization.

The rest of this paper is organized as follows. In Chap. 2 we briefly review some of the previous work that has been done in web-based Molecular Visualization. Chapter 3 gives a short description of common molecular models, corresponding computation algorithms and molecular properties. In Chap. 4 we outline our approach and Chap. 5 presents our results. Finally, we give our conclusions in Chap. 6 and point out promising future research.

2 Previous and Related Work

Within the scope of the World Wide Web special description languages for chemical objects apart from HTML have been defined (e.g. CML[1]). Moreover, for several molecule data formats (e.g. PDP, XYZ, Alchemie) extensions of the MIME format exist, requesting the use of special Browser plug-ins for visualization (e.g. Chime[2]). For the present the most common description format to distribute three-dimensional data over the World Wide Web is certainly VRML,

[1] the Chemical Markup Language, see http://www.venus.co.uk/OMF/.
[2] see http://www.mdli.com/download/chime/.

which may be replaced by something like Java3D in future. [1] describes a general WWW-based visualization service where the user enters some data on the client which is transferred to the server calculating the VRML model which is then retransmitted to the client for display. Each shape changing interaction by the user starts another cycle. The "Virtual Reality Modelling Language in Chemistry"[3] fits to this description. A complex in-house chemical information system based on Internet programming techniques with selected tasks shifted to the client's side is described in [2]. [3] presents a Java applet for interactive 3D visualization on the Web which makes use of the Marching Cubes Algorithm on the client's side. In [4] a progressive approach for transmitting Isosurfaces to an applet is applied to avoid Internet bottlenecks. The Molda System [5] is a client-side molecular modeling and graphics system using Java and VRML and offering the visualization of standard molecular models.

What all these systems have in common is that they emphasize the server's role (clients are pure display clients) or that they do not exploit the clients' today's abilities.

3 Molecular Models and Properties

3.1 Standard Models

The most common standard models in Molecular Visualization are Wireframe Models, Stick Models, Ball-and-Stick Models and space-filling Corey-Pauling-Koltun (CPK) Models (see Fig. 2 for examples).

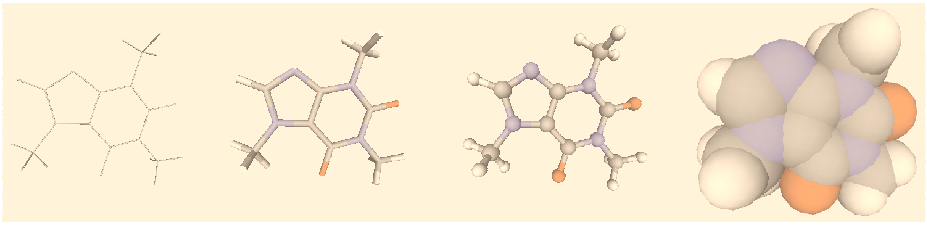

Fig. 2. Standard molecular models *(from left to right: Wireframe, Stick, Ball-and-Stick and CPK Model of caffeine)*

In *Wireframe Models* bonds are represented by lines while atoms are not regarded. *Stick Models* extend this idea by representing the bonds by cylinders of a chosen diameter. *Ball-and-Stick* Models include atoms represented by spheres which diameters are a chosen fraction of the atoms' Van-der-Waals radii. In *CPK Models* bonds are invisible with atoms visualized as spheres using their Van-der-Waals radii. All standard models choose their colors according to the CPK standard colors.

[3] see http://www.pc.chemie.tu-darmstadt.de/vrml/.

3.2 Surface Models

Computerized visualizations allow the display of various molecular surfaces, which depend on positions and radii of the atoms forming the displayed molecule and on chemical or physical properties. In addition, different molecular properties can be mapped onto these surfaces. A comparison of common molecular surface definitions can be seen in Fig. 3.

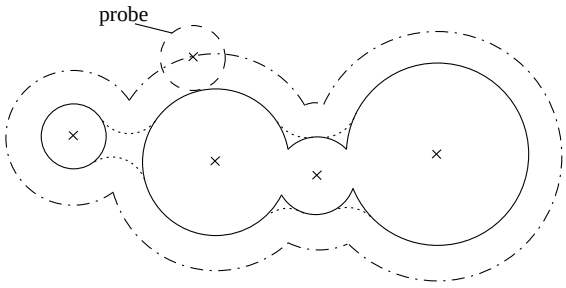

Fig. 3. Comparison of different molecular surface definitions *(solid: Van-der-Waals Surface, dot: Richards' Contact Surface, dash-dot: Richards' Accessible Surface)*

The *Van-der-Waals Surface* is the simplest molecular surface. It consists of the visible part of the union of all atoms composing the molecule displayed with their Van-der-Waals radii.

Richards' Contact Surface [6] is created by rolling a probe sphere of a given radius, usually 1.4 Å (sphere including a water molecule H_2O), over the Van-der-Waals Surface. The surface is composed of two kinds of surface patches: the part of the Van-der-Waals Surface of each atom which is accessible to the probe sphere and the inward facing part of the probe sphere when it is simultaneously in contact with more than one atom. Richards' Contact Surface is constructed using Conolly's Algorithm [7, 8] combining an analytical computation of the different surface patches with a following triangulation step of a chosen accuracy. The previously described Van-der-Waals Surface can be treated as a special case of Richards' Contact Surface using a probe sphere of infinitesimal radius.

Richards' Accessible Surface [6] is constructed in the same way as Richards' Contact Surface, except that the surface itself is composed of the positions reached by the probe's center.

Isosurfaces represent all spatial positions with constant values of a chosen scalar molecular property. A very popular way to compute an Isosurface is the Marching Cubes Algorithm [9]. After a grid based subdivision of the given volume of density values this algorithm merges local triangular approximations to an approximation of the entire Isosurface. Of course, Isosurfaces depend on the calculation method used for the selected molecular property.

3.3 Molecular Properties

There are a lot of different molecular properties which can be mapped onto the previously described molecular surfaces or which can be used to define Isosurfaces. Molecular orbitals are generally represented by Isosurfaces. Curvature can describe the curvature of a given molecular surface, the gradient of a selected scalar property or the value (of the gradient) of a non-scalar molecular property. Color mapping of interaction energy allows the chemist to rapidly and intuitively locate the regions corresponding to favorable interactions. Hydrophobicity represents the molecule's aversion against water molecules. It could be approximated using the tables found in [10]. A simple approximation of the molecule's electrostatic potential V is given by

$$V(r) = \sum_{i=1}^{n} \frac{q_i}{|r - r_i|} \tag{1}$$

where r is the selected position, r_i the atoms' centers and q_i the atoms' electronegativities.

4 Our Approach

Our goal was the design of a web-based Molecular Visualization system meeting the following requirements: 1. The system structure has to allow easy extension and exchange of system components. 2. The construction of the system has to provide a way to load the client's components dynamically over the Internet. 3. The system should not be affected by Internet or server bottlenecks. 4. The system has to operate platform independent but it should make use of resources offered by the client machine (e.g. OpenGL hardware acceleration). 5. The chemical data to visualize has to be transferred via the Web in a common standardized data format. 6. The system should offer advanced molecular surface models with real-time interaction rates. 7. The Graphical User Interface has to provide a straightforward and standardized user control.

The stated requirements lead quickly to the use of the Java programming language guaranteeing perfect integration with the World Wide Web and excellent platform independent use. Additionally, the object-oriented structure of Java permits the split of the system into interacting components and allows easy maintenance. Besides this, the ability to produce standardized Graphical User Interfaces is one of Java's inherent properties. All remaining requirements will be the contents of the following paragraphs explaining the system's structure and its interacting components with their functions.

4.1 Environment

The entire system has been developed using Java version 1.1. For the reason of Java3D not yet running reliably enough for our tasks we have chosen VRML 2.0

as an intermediate solution. The display of the VRML models is done by Silicon Graphics' CosmoPlayer, which is available as a Browser plug-in for Windows based systems, for SGIs and for Macintosh platforms. For the communication with the VRML Browser the External Authoring Interface (EAI) is used. We apply Netscape's Navigator as our favorite Web Browser.

On the server's side arbitrary http servers providing a Common Gateway Interface can be used – we are applying Apache and Roxen Challenger servers.

We use the Brookhaven Protein Data Bank (PDB) Format[4] as a standardized molecule data format. This is no restriction with Babel[5] offering on-line translation between different data formats.

4.2 System Structure

The design of our system follows the client-server approach but with essential tasks dedicated to the clients. Figure 4 shows a simplification of the system's design.

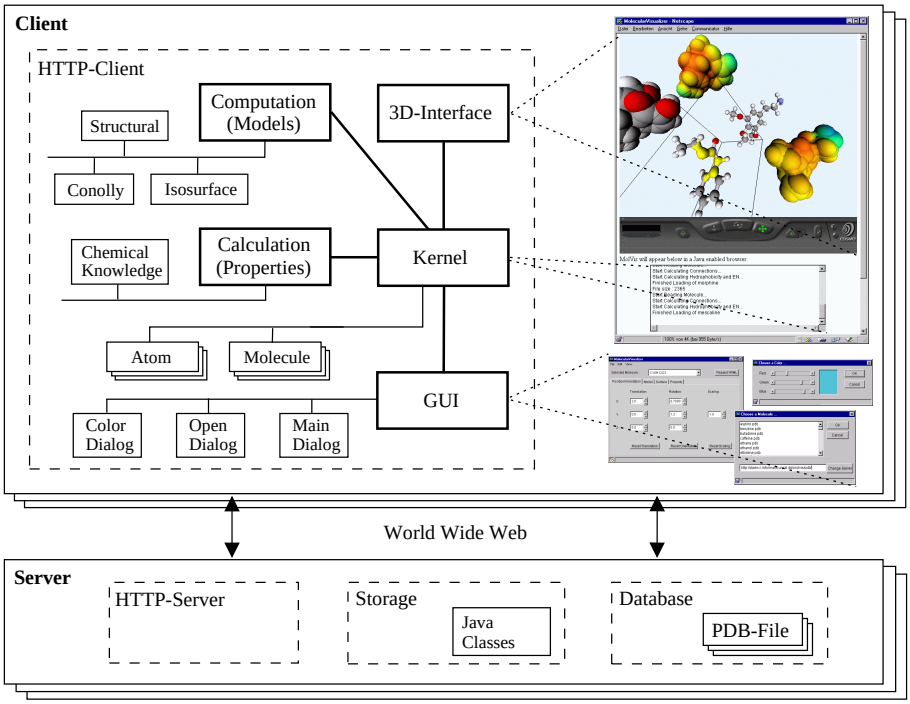

Fig. 4. System structure

[4] see http://pdb.pdb.bnl.gov/.
[5] see http://www.chemoinformatics.com/free/babel.html.

The server's side The server machine runs a http server daemon which can access a local database of chemical atomic coordinate files in PDB format. The http server has to provide a list of PDB files and it has to be able to send each PDB file in a compressed format to a client. The client's Java classes are stored on a server to take advantage of dynamic Java class loading.

The client's side The client machine has to run a Java enabled http client, usually a Web Browser. The client's part of our system consists of five main components each corresponding to one or more Java packages, but we will disregard single Java classes and the Java class hierarchy in our descriptions.

The *kernel* corresponds to the Java applet which actually is the most important part of the Java program on the client's side controlling the main thread and establishing all connections with the user's Web Browser and with the 3D-interface. The kernel initializes the other modules and it is responsible for all internal and Internet communication. The client's part of the Molecular Visualization system can handle several molecules with individual visualization parameters and molecular models at the same time – one of them is always labeled as the active molecule. For this purpose each molecule is stored separately, including all information concerning it (atoms, current molecular models, visualization parameters). Operations (add, delete, duplicate) on this internal database for molecules are also handled by the kernel component. It is important to point out, that all data concerning visualization is stored in a native format to become independent of the currently used 3D-interface and its data format. The kernel displays all kind of status information (molecule name, communication status, computation and calculation progress, memory usage, et cetera) in a console window below the 3D-interface in the user's Web Browser.

The *computation module* is responsible for the production of visualization-ready molecular models which can be passed through the kernel to the 3D-interface. While the computation of a standard model is a simple generation of drawing primitives the computation of an advanced surface model requires time and memory consuming algorithms. We have implemented Conolly's Algorithm to compute Richards' Contact Surface. The accuracy in the triangulation step is determined by a user-controlled error criterion. Isosurfaces are computed with the Marching Cubes Algorithm. In addition to a resolution vector (n_x, n_y, n_z) used for the subdivision of the regular grid the user can limit the algorithm's processing to a bounding box. This is extremely useful to examine surface details. The computation module is also responsible for the mapping of molecular properties onto surfaces models.

The *calculation module* offers algorithms approximating molecular electrostatic potential and hydrophobicity at any given spatial position. Currently both approximations work with weighted sums so the calculation time is proportional to the number of atoms in a molecule.

The *3D-interface* is responsible for the display of the molecular models. It provides drawing primitives like lines, spheres, cylinders or triangle meshes representing complex surface shapes. The primitives' properties, especially color, are

used to map molecular properties. Besides navigation (rotating, moving, zooming) with the mouse, our current 3D-interface offers simple user interaction with callback functions. The 3D-interface is supposed to exploit the client's graphical power to maximize the visualization quality, for example frame and interaction rate. Silicon Graphics' CosmoPlayer for instance is able to use the support of hardware accelerated OpenGL graphics boards on Windows based systems. This component encapsulates all accesses to the three-dimensional in- and output, so it can easily be adapted to Java3D in future.

The *GUI* is responsible for all user interactions. Its dialog windows offer various possibilities to start up actions (e.g. calculations) and to change parameters.

Networking A client can receive PDB data from several servers on the Internet and on the other hand one server may offer its service to many clients on the Web. Internet connections are only kept for short times when they are really needed. This is the case for the dynamic Java class loading, for the choice of a molecule from a server's browsable list and for the download of a molecule's PDB description. Java classes as well as molecular descriptions are both transferred over the Web in a compressed format to avoid bottlenecks.

5 Results

To examine our system's capabilities - in particular its computational and its rendering performance - it has been tested on different computer platforms (Windows and SGI based systems with different hardware configurations) with several molecules of different structure and complexity. As an example we present some results achieved on a today's typical client system, a medium equipped Windows based PC System with a 350MHz Pentium II processor, 128MB main memory and a hardware accelerated OpenGL graphics board – currently such a configuration can be obtained for less than $US 1,000. The speed analysis in Table 1 lists the measured computation time for the model, the resulting triangle count to be displayed and the frame rate achieved. The numbers indicate a minor importance of speed restrictions introduced by the Java programming language. Note that the Ball-and-Stick and the CPK Model do not use triangle primitives.

Table 1. Measured computation time (sec), triangle count and frame rate

	Ball-and-Stick			CPK			Richards' Contact			Isosurface		
ethanol	0.0	–	30.3	0.0	–	29.9	1.559	2,668	20.5	5.815	22,690	6.1
aspirin	0.0	–	20.1	0.0	–	19.1	4.592	10,922	11.9	6.423	22,492	6.2
morphine	0.0	–	9.5	0.0	–	8.2	14.942	34,098	6.2	46.387	18,436	9.2
strychnine	0.0	–	8.4	0.0	–	7.0	19.311	43,502	4.6	55.739	18,414	8.7

In the upper left picture of Fig. 5 (see Appendix) a Ball-and-Stick Model of ribonucleic acid is combined with an Isosurface (yellow, transparent) determining the value 140 of electrostatic potential. In the upper right picture hydrophobicity is mapped onto a Richards' Contact Surface of vitamin-b1 while the Isosurface (grey, transparent) depicts a hydrophobicity value of 0.7. The same Isosurface can be seen in the Stick Model of vitamin-b1 with electrostatic potential mapped onto its surface. A second Isosurface (green) determines the value 2.0 of hydrophobicity. The restriction of Isosurface calculations to a bounding box is very useful for complex molecules, for example the plant seed protein in Fig. 1 (see Appendix). Here electrostatic potential is mapped onto an Isosurface depicting a hydrophobicity value of -1.7 while a second Isosurface (purple, transparent) determines a hydrophobicity value of -2.2. Both Isosurfaces are restricted to a chosen volume respectively.

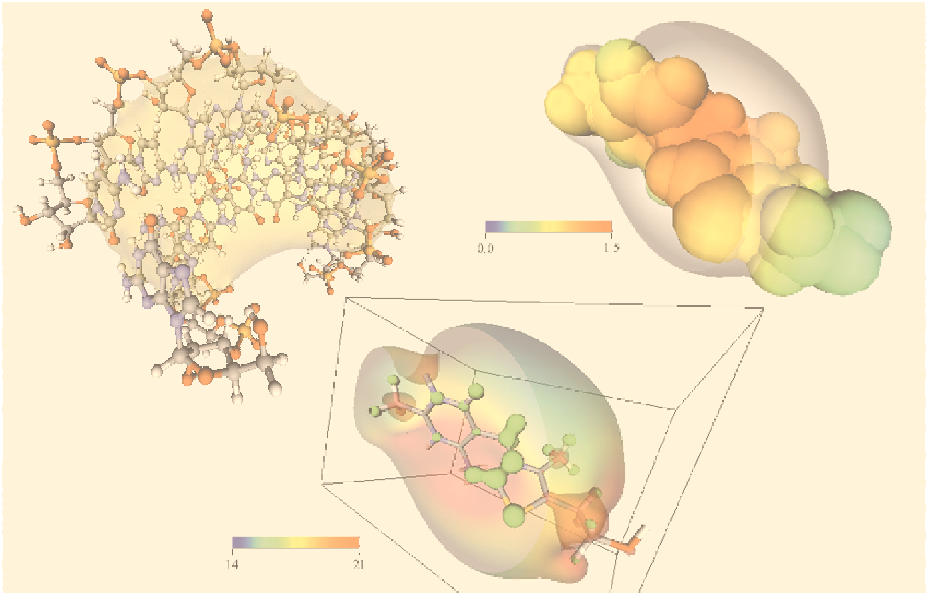

Fig. 5. Ribonucleic acid *(Ball-and-Stick Model, Isosurface)*, vitamin-b1 *(Richards' Contact Surface, Isosurface)*, vitamin-b1 *(Stick Model, different Isosurfaces)*

6 Conclusion and Future Work

Our idea of an efficient, client-side Molecular Visualization system offers much more interaction to the user than standard web-based client-server systems. Although the use of the Java programming language implies a reduced execution speed in some areas, Java leads to a flexible program design allowing easy extension and providing excellent portability. In fact, according to our performance

tests computational and graphical power is nothing to be really concerned about any longer. Furthermore the users are offered several parameters to regulate the trade-off between visualization quality and time. Our system offers the visualization of various molecular models and properties which can be displayed at the same time. By placing all necessary calculations on the client's side our approach exploits the computational power of modern desktop workstations and personal computers. This also reduces the needed Internet bandwidth, only a few kilobytes of data containing the molecule description are transferred – the users become more independent of server or Internet bottlenecks.

With regard to the still decreasing costs of computational power and the further development of the World Wide Web, this powerful approach promises new possibilities in teaching, tele-working, Web publishing or even drug design. Thinking about the new generation of Network Computers (NCs), offering hardware Java virtual machines, for many tasks of web-based (Molecular) Visualization, run time reasons calling for specific applications and hardware platforms will not remain the killing argument in future.

Currently we are working on the exchange of our present 3D-interface to Java3D. This will finally remove the last platform dependence of our system. Online editing of molecules and the calculation of the following client-side re-evaluation of the molecules is one topic of future research.

References

1. Trapp, J. C., Pagendarm, H.-G.: A Prototype for a WWW-based Visualization Service. 8th Eurographics Workshop on Visualization in Scientific Computing, Boulogne sur Mer, (1997).
2. Ertl, P. (Novartis Crop Protection AG): Molecular Modelling through the World Wide Web. Chemistry and the Internet Conference, (1998).
3. Michaels, C., Bailey, M.: VizWiz: A Java Applet for Interactive 3D Scientific Visualization on the Web. IEEE Visualization, , (1997) 261–267.
4. Engel, K., Grosso, R., Ertl, T.: Progressive Iso-Surfaces on the Web. IEEE Visualization, Late Breaking Hot Topics, (1998) 37–40.
5. The Molda System: Molecular Modeling and Molecular Graphics using VRML Viewer. See http://cssj.chem.sci.hiroshima-u.ac.jp/molda/.
6. Richards, F. M.: Areas, Volumes, Packing, and Protein Structure. Annual reviews of Biophysics and Bioengineering, Vol. 6 (1977) 151–176.
7. Conolly, M. L.: Analytical Molecular Surface Calculation. Journal of Applied Crystallography, Vol. 16 (1983) 548–558.
8. Conolly, M. L.: Molecular Surface Triangulation. Journal of Applied Crystallography, Vol. 18 (1985) 499–505.
9. Lorensen, W. e., Cline, H. E.: Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics, Vol. 21 (1987) 163–169.
10. Viswanadhan, V. N., Ghose, A. K., Revankar, G. R., Robins, R. K.: Atomic Physiochemical Parameters for Three-Dimensional Structure-Directed Quantitative Structure-Activity Relationships IV. Additional Parameters for Hydropobic and Dispersive Interactions. Journal of Chemical Information and Computer Science, Vol. 29 (1989) 163–172.