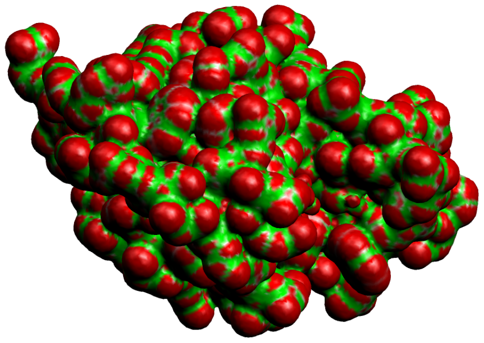
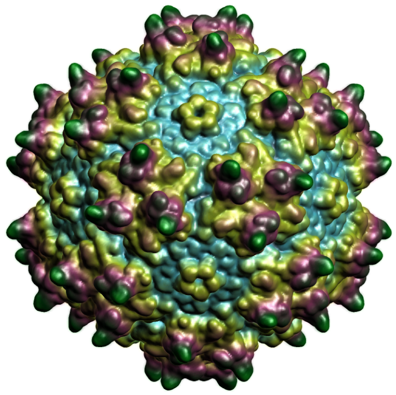
TexχMol User Guide





September 20, 2007

Computational Visualization Center
Institute for Computational Engineering and Sciences &
Department of Computer Sciences
The University of Texas at Austin
http://www.ices.utexas.edu/cvc

Contents

Contents **2**

Preface **5**

1 Introduction **7**

1.1 Functions . . . 8

1.2 Installation . . . 8

1.2.1 Step by step installation guide . . . 9

1.2.2 Common installation problems . . . 9

1.3 The main user interface . . . 10

2 Opening and rendering files **13**

2.1 Loading a molecule description file . . . 13

2.1.1 The static hierarchy for biomolecules . . . 13

2.2 Loading volume files . . . 16

2.3 Loading surface files . . . 16

3 Computations **19**

3.1 Isosurfaces . . . 19

3.1.1 Rendering isosurfaces . . . 19

3.2 Curvatures . . . 19

3.3 Electron density . . . 20

3.4 Hydrophobicity . . . 23

3.5 Contour spectrum . . . 23

3.6 Pocket . . . 26

3.7 Curation . . . 26

3.8 Secondary Structure Elucidation from 3D Maps . . . 28

3.9 Meshing of Molecules . . . 28

3.10 Protein-Protein Docking . . . 29

4 Scripting and animations **33**

4.1 Batch-mode calculations . . . 33

4.1.1 Electron density and hydrophobicity . . . 33

4.1.2 Curvatures . . . 34

4.2 Scripting 35
4.3 Animations 36
4.3.1 Creating a trajectory 36
4.3.2 Saving animations 36

Acknowledgements **39**

Bibliography **41**
4.4 License Agreement 43

TexMol

This is the user documentation for *TeχMol* v1.0. This is to help both new users familiarize themselves with the software, and to provide a comprehensive list of functions to everyone. There is a separate programmer documentation available for those interested in extending the functionality of *TeχMol* , which is an open source software. *TeχMol* was developed at the Center for Computational Visualization under Dr. Chandrajit Bajaj, at the University of Texas at Austin.

Chapter 1

Introduction

Computational visualization of large molecules – particularly for protein and RNA structures – has gained tremendous importance as a cutting-edge tool for biological research. Previous work has focused on efficient rendering for single-component, static molecules, which is becoming increasingly restricted in light of the increasing demand for more complex, dynamic visualization and representation. We present *TeχMol* (short for Texture Molecular Viewer), an interactive molecular exploration package created in response to the increasingly demanding visualization needs of the biology community.

To efficiently visualize dynamic and flexible structures, *TeχMol* uses a molecular specification file to construct the Flexible Chain Complex (FCC), a robust, dynamic data structure that serves as *TeχMol* 's internal representation for molecular structures. The FCC models the flexible joints of a molecule and contains a biochemically-based hierarchy for level-of-detail optimizations.

Besides high visualizer functionality, *TeχMol* delivers rapid, accurate rendering via various novel applications of texture-based rendering techniques for structural and for volumetric representations. For the field of structural representation (e.g. CPK (of union of balls, where each atom is represented by a sphere with the radius equal to the van der Waals radius.), ball-and-stick model), recent advances in programmable graphics hardware have opened the door for texture-based rendering – also known as imposter rendering – that greatly reduces geometric complexity while preserving, and in some cases improving the visual fidelity of the final image. *TeχMol* 's level-of-detail hierarchy allows for static and dynamic multiresolution, which reduces the visual clutter that often accompanies atom-level visualization while still maintaining biochemical structural information, such as residue-level grouping. Combined with the level-of-detail hierarchy, texture-based rendering allows *TeχMol* to render large and previously intractable molecules.

TeχMol also supports efficient volumetric visualization via texture-based techniques. By combining rendering modes, the visualizer can either map volumetric data onto the structural model of the molecule or it can juxtapose multiple volume sets and structure models concurrently. In both cases, the resulting visualization ties molecular structure to molecular function in an elucidating manner.

1.1 Functions

TeχMol is both a visualization and computational toolkit for large biomolecules. Some of the main features include

- Open source software for rendering large molecule data sets. It has been tested on molecules with more than 3million atoms.
- Written in C++, with a QT front end. The same code should work on multiple platforms, including Windows and Linux.
- Reads in the well known PDB and PQR formats.
- Produces high quality images using High-end graphics cards functionalities.
- Volume rendering and calculation of electron density, hydrophobicity functions.
- Wireframe and smooth shaded isosurfaces.
- Computes metrics including curvatures, surface areas and volumes.
- Provides programmers with a simple and yet powerful hierarchical data structure of the molecules, with various torsion angles calculated.
- Multiple views and multiple data set rendering.

1.2 Installation

Please install *TeχMol* and perform the functions described in the rest of the tutorial to learn more about the software. *TeχMol* is open source and should be available for download from CCV's software download page. It has been tested under both windows and linux. You will need the following

- A windows or linux operating system.
- QT and a C++ compiler.
- A high end graphics card, like GeForce fx or better.
- An input PDB file, which can be downloaded off the PDB database.

Once you have downloaded the source, you can compile it using either the QT's Makefiles (the .pro files) or the project workspace files (for windows).

1.2.1 Step by step installation guide

There are some subtle problems which could come up when compiling and linking *TeχMol* . Here are some solutions.

- Check to ensure that you have a good graphics card. We have currently tested our programs on NVidia's cards, GeForce fx and beyond.
- Upgrade your graphics card drivers and test example Cg programs they usually provide. There should be examples on NVidia's website, which can help you ensure that it is set up well.
- Make sure that the latest Cg compiler is installed.
- Define the environment variables CG_BIN_PATH, CG_LIB_PATH and CG_INC_PATH if it was not already done.
- Download and install QT. There is a free version for Linux.
- Make sure that the variables QTDIR and QTLIB are defined.
- If the contour library is being used, the user, unfortunately needs to define whether they are using a big endian or little endian machine. Hence, you may or may not need to define the variable LITTLE_ENDIAN. Other libraries do not need this and can figure out the endianness in the code.

1.2.2 Common installation problems

- Compilation problems
  - *QT and X11 incompatibility*
    You could get into problems due to QT and X11 defining things arbitrarily. One way to get around this is to change some headers or redefine the error causing terms.
  - *Gl extensions*
    Download the latest glext.h from SGI's website. DELETE other vendors glext.h. Before installing *TeχMol* , check to see if Gl and Cg demos are working.
  - *file not found*
    QT's internal compilers may not have created the files needed by a library in a subfolder of *TeχMol* . Try compiling a different subfolder or the main folder itself and try again.
- Rendering issues
  - *Doesn't render*
    Cg files provided are linked in at run time. So you will need to keep them in the same folder as *TeχMol* 's executable.

1.3 The main user interface

In figure 1.2.2, we show a volume rendering and isosurface of a molecule. The different regions in the user interface are

- Molecule browser. The list of molecules are shown here. The name is displayed, and if it was left blank on opening the file, the full filename is shown. The user can select files to either manipulate it using the properties window or to delete it.

- The rendering area. This is either in perspective or orthographic mode, runs using OpenGL and can be either a single or multiple views.

- Properties area. Each type of file (volume, molecule or surface) has its own unique properties widget which is displayed here. On selecting a data set from the browser, its corresponding properties are displayed here.

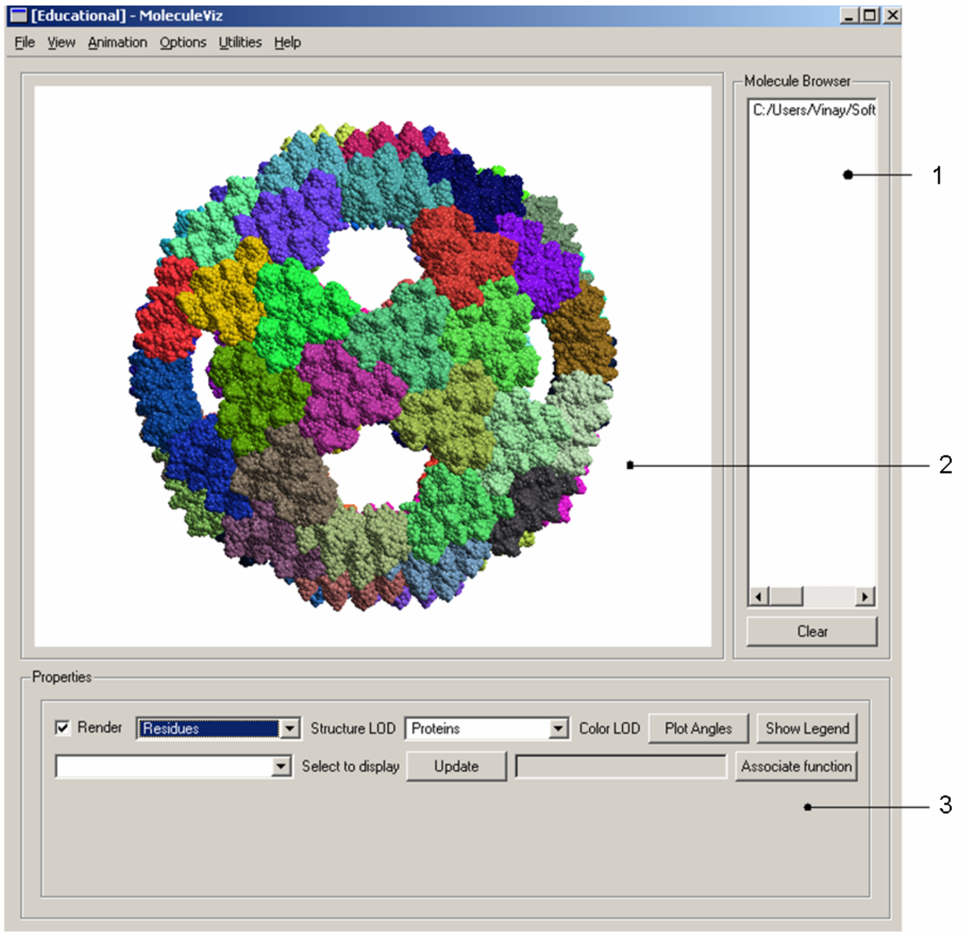


Figure 1.1: The main user interface of *TeχMol* . In this figure, we mark the three main regions users should get familiar with: 1. *Molecule browser* where data sets names are displayed 2. *Rendering area*, where data is rendered and 3. The *Properties widget* where the properties of the selected data set is shown.

Chapter 2

Opening and rendering files

TexMol reads molecule-specific PDB, PQR files (including simpler XYZ, XYZR, PTS and a custom GOA formats), volume files (RAWIV, RAWV, DX, MRC), surface files (RAW, RAWC, RAWN, RAWNC, OBJ, C2C) and a custom NURBS file. By default, the data set is neither centered to the view, nor rendered to screen. The user needs to specifically render the data set after loading it.

Warning:If you do not have a good high end graphics card, or if it is not set up properly to display using Cg, you will not see any output on the screen. We are in the process of moving to OpenGL Shading Language.

2.1 Loading a molecule description file

We support multiple molecular files. The most commonly used file formats are PDB [2] and PQR [3]. We also have simple file formats to describe general point sets: XYZ (list of centers on successive lines), XYZR (same, includes radius next to center) and PTS (similar to XYZ, but first line contains number of centers). The properties box for molecular data sets is shown in figure 1.2.2. There are two LODs as shown in the figure, the color LOD and the object LOD.

2.1.1 The static hierarchy for biomolecules

We use the biochemical hierarchy in our visualizations. For atoms, we use the CPK model, with van der Waals radii. The colors and radius used are described in the table in *elementInformation.h*. Atoms are grouped into residues, which are approximated by spheres in the structure LOD. Secondary structures are rendered with imposters like cylinders and helices. Chains are rendered as a ball-and-stick model.

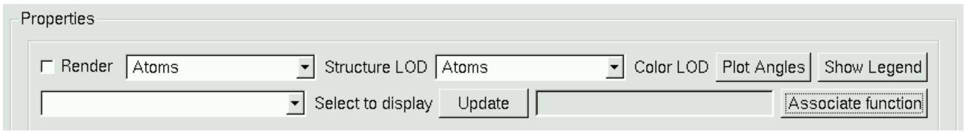


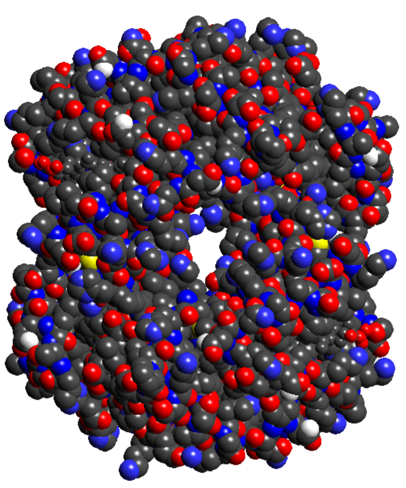
Figure 2.1: The *Properties widget* for molecule data sets allows users to pick their choice of color and structure level of detail to render. Other functions include associating a volumetric function with the surfaces color.

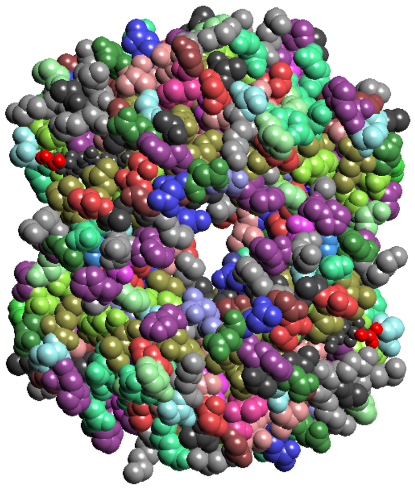
- Atoms. The atom sequence is the lowest, or finest level in the hierarchy to obtain the primary structure of the molecule.
- Residues. Atoms are grouped into their respective residues.
- Secondary structures. Residues are grouped in to either helices or sheets, or into dummy NULL structures.
- Chains. The secondary structures are collected in to chains. They form the tertiary structure.
- Molecule. The highest or coarsest level in the static hierarchy is the molecule itself.

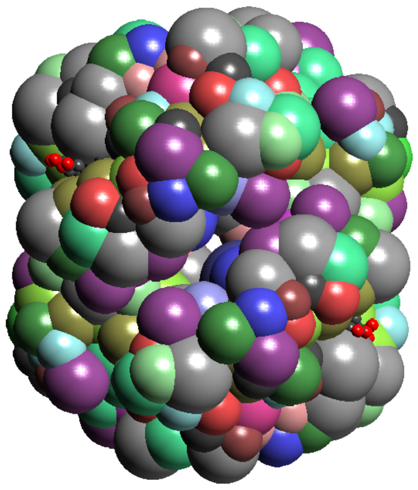
After loading the molecule file, one can select a combination of color and structure LOD to obtain various visualizations. To open a PDB or PQR file, follow the steps shown in table 2.1. A simple well known molecule to start with would be hemoglobin (1A00.pdb), as it has multiple chains and helices. Some interesting combinations of visualizations would be as shown in figure 2.2. Table 2.2 explains how one can visualize the helices in the loaded molecule.

Table 2.1: Loading a molecule

- Select File - Open from menu bar.
- Choose a file name by pressing the File button and select a PDB or PQR file.
- If you know that you are choosing a isosahedral virus, check the isVirus check box.
- Press OK to load the file, or Cancel to cancel this action.


(a) Atoms colored with element colors


(b) Atoms colored with their respective residues colors


(c) A coarser model with residues rendered as spheres

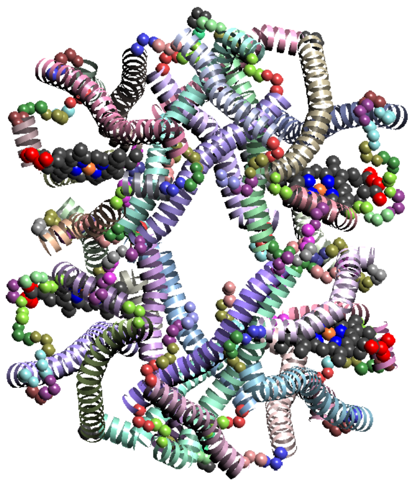

(d) Helices forming the secondary structures in the molecule

Figure 2.2: Molecules can be rendered at different color and structure level of details. Here we show the hemoglobin molecule (1A00.pdb) rendered in 4 different styles.

Table 2.2: Visualizing helices in a molecule

- If you have not loaded a molecule, do so by following the steps outlined in Table 2.1.
- Set the Structure LOD to Secondary structures and the Color LOD to any item.
- Render the molecule by checking the Render check box.
- If the molecule is not visible in the current view point, you may need to zoom out and translate to bring it in to view.

2.2 Loading volume files

We currently support four volume files including the scalar RAWIV, MRC and DX formats, and vector valued RAWV formats. They are both binary and ascii files and support multiple data types and sizes. For a complete description, see the file format descriptions on the CCV web site. To load volume files, follow the same steps as outlined in table 2.1, but select a volume file instead of a molecule file.

To render volume files and isosurfaces, you need to get familiar with a widget called a color table, which is user interface for reading a transfer map. A detailed description for the color table can be found in the volume rover software user guide.

2.3 Loading surface files

Warning:Surface files currently supported can be compressed or not. The uncompressed files can take up to a few minutes to load for very large meshes.

Like volume files, surface files do not need to deal with molecules. *TeχMol* currently supports uncompressed (RAW, RAWC, RAWN, RAWNC and OBJ) and compressed (c2c) surface files. To load any of these file formats, table 2.1 can be followed with the right input file types. Both wireframe and surface rendering modes are supported. The user can also view both together. A visualization of the Mache molecule, showing the gorge is shown in figure 2.1.1. The OBJ file format is from Alias Wavefront.

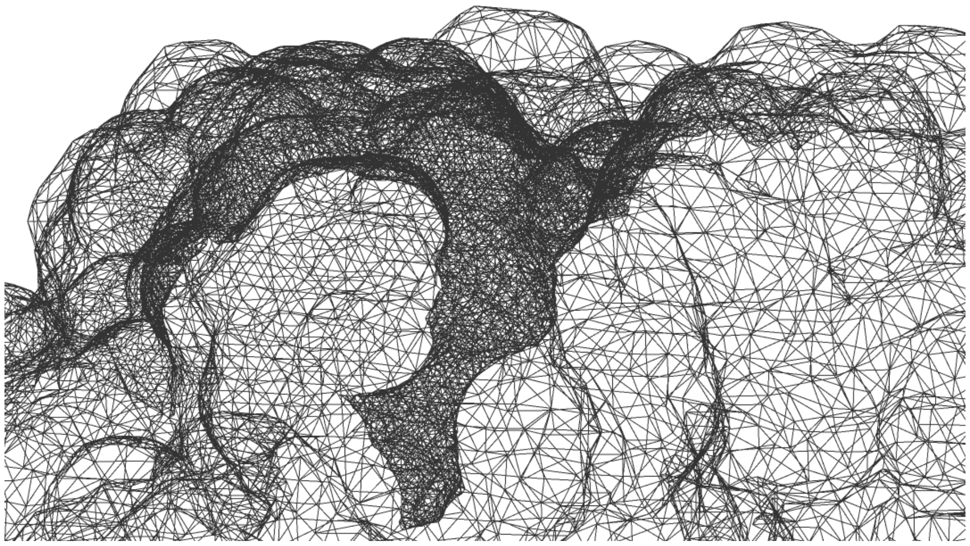


Figure 2.3: Wireframe rendering of a mesh, showing a gorge type feature in the surface of the MACHE molecule.

Chapter 3

Computations

Apart from visualization, *TexMol* also allows users to compute functions of the molecule including surfaces and surface functions like isosurfaces, curvatures, and the contour spectrum and volume functions like electron density and hydrophobicity.

3.1 Isosurfaces

Given a volume $f(x, y, z)$, an isosurface with isovalue c is defined as $f(x, y, z) = c$. The extraction of isosurfaces from volume files is known as isocontouring. There are two packages for isocontouring within *TexMol* . One is the *marching cubes* algorithm implemented in the *Contouring* library and the other is the *seed set* algorithm implemented in the *contourlib* library. To create an isosurface, the user needs to load in a volume and extract it at a given isovalue as described in table 3.1.

Warning:Isosurfacing of large volume files could take up to a minute.

3.1.1 Rendering isosurfaces

Isosurfaces are rendered by default on creation and cannot be hidden. We also currently use smooth shading for rendering the meshes. The color of the mesh at any vertex is given by the color of the volume at that point. The color of the entire mesh can be changed by right clicking on the isocontour bar and selecting edit.

3.2 Curvatures

Using a functional definition for the electron density representation of a molecule, the curvatures at any point can be estimated. Analytical definitions

Table 3.1: Isosurfacing

- Load a volume if it is not already done and select it in the molecule browser to bring up its property widget.
- In the color table, right click at any point and add an isocontour.
- This could take a few seconds to a minute or so.
- The isovalue can be modified by dragging the isosurface bar in the color table.
- The isosurface can be deleted by deleting its isosurface bar.

of the functions yield analytical equations for deriving the curvatures. *TexMol* estimates two curvatures, the Mean curvature H and the Gaussian curvature K. If k_{min} and k_{max} are the minimum and maximum curvatures at a point, then

$$H = \frac{1}{2}(k_{min} + k_{max}),\ K = k_{min} \times k_{max} \tag{3.1}$$

These are calculated from a potential function ϕ as follows

$$H = \frac{C(f_x^2(f_{yy} + f_{zz})) - 2C(f_x f_y f_{xy})}{2(C f_x^{2^2})},\ K = \frac{2C(f_x f_y (f_{xz} f_{yz} - f_{xy} f_{zz}))}{(C(f_x^2))^2} \tag{3.2}$$

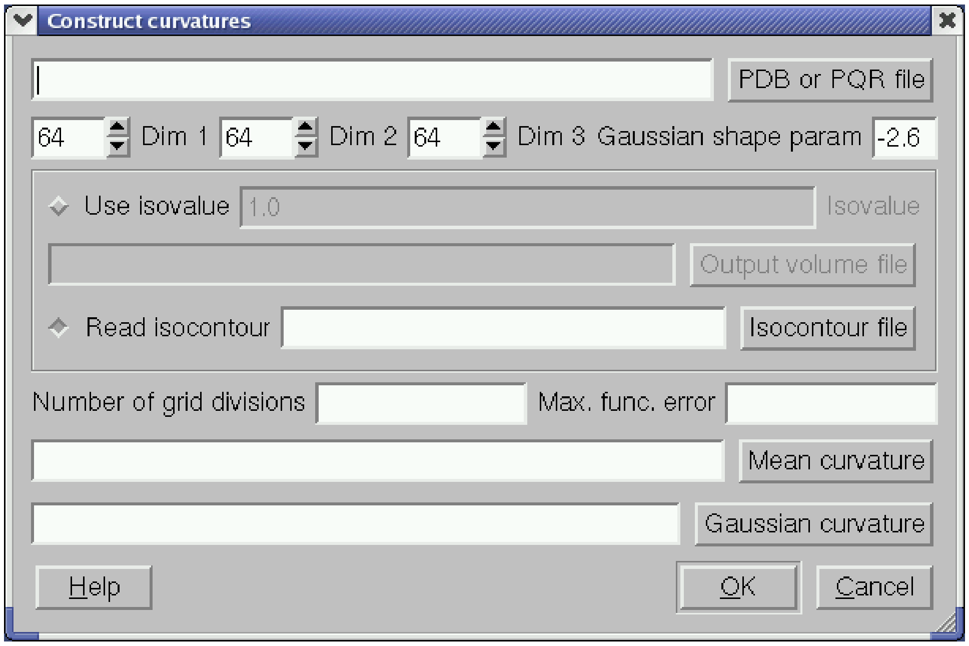
C denotes cyclic summation over x, y, z,
subscripts denote partial differentiation with respect to those variables.

There are two ways in which most functions can be calculated in *TexMol* . First, the user can do it through the graphical user interface. Second, it can be done in batch mode. In table 3.2, we describe how to calculate the mean and gaussian curvatures from the graphical user interface.

3.3 Electron density

The electron density has been used as a description of molecular shape. One commonly used model of sum of radial functions has been used in *TexMol* . This field can be described at any point x, y, z in a volume as a scalar value given by

$$\phi_{dens} = \sum_{i=1}^{M} e^{\beta_i} \tag{3.3}$$



64 Dim 1 64 Dim 2 64 Dim 3 Gaussian shape param -2.6

Use isovalue 1.0

Read isocontour

Number of grid divisions

Max. func. error

Figure 3.1: This curvature widget can be opened by selecting Utilities - Construct curvatures from the main menu bar.

Table 3.2: Estimation of curvatures from GUI

- Consider figure 3.1.1 for this example.
- Load a molecular file to estimate curvatures for.
- Since we need an electron density function definition, select values for the volume dimensions and the gaussian blobbyness parameter.
- If you know the points at which to estimate the curvature, you can load the mesh file, or ask *TeχMol* to perform isocontouring at some selected isovalue.
- The number of grid divisions and the function error allowed are tradeoffs to performance.
- Three files, the mean and gaussian curvature, and the curvature values themselves are written out to the selected files.

and

$$\beta_i = blobby * \frac{(x - xc_i)^2 + (y - yc_i)^2 + (z - zc_i)^2}{vr_i^{\,2}} - blobby \tag{3.4}$$

M	is the total number of atoms,
xc_i, yc_i, zc_i	is the center of the i^{th} atom,
vr_i	is the van der Waal radius of atom i and
$blobby$	is a parameter which controls the shape of the gaussian.

Using the graphical user interface, the users can compute the electron density of a molecule as shown in table 3.3. See [4] for a fast summation algorithm. There are four types of electron density volumes we compute with respect to visualization.

1. Scalar valued electron density.
2. Vector valued electron density, where there are 4 tuples at each point. RGB at a point contains a color value, and A contains the density itself. This can be computed in three ways. See figure 3.2 for an example.
   (a) Color according to a structure like atoms, residues etc.
   (b) Color using the specifications in a color map file.
   (c) Use depth coloring to bring out surface features. This feature currently takes as input a volume file (scalar or volume). Load a

volume file and select Utilities, Construct depth colored volume from the menu bar.

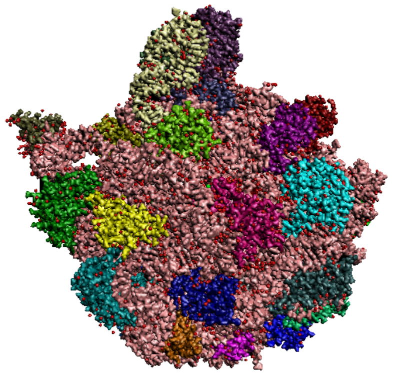
Table 3.3: Electron density computation

- Load a molecule file if it is not already loaded.
- Select the molecule from the molecule browser.
- Select Utilities and Construct volume from the menu bar.
- Select Electron density as the function needed to be computed and give the dimensions and blobbyness parameter.
- If you need a output file, then enter a output file name, or just leave it blank.
- RawV volumes or vector valued volumes can also be constructed by choosing it in the check box.
- If rawV was chosen, then also choose a color LOD for the density.
- The generated volume can be loaded in to memory or not by checking Load generated volume.

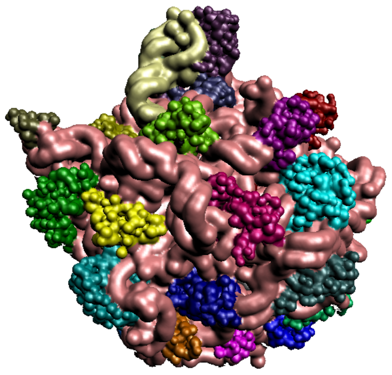
3.4 Hydrophobicity

Hydrophobicity maps can be created in a similar fashion as rawiv electron density files. Follow the procedure outlined in table 3.3, but choose hydrophobicity instead of electron density. You probably cannot create anything meaningful by choosing hydrophobicity and color mapped volumes together. The hydrophobicity is given per atom as shown in the table in file *elementInformation.h*. The hydrophobicity of the dengue virus capsid protein (1r6r.pdb) is shown in figure 3.1.1.

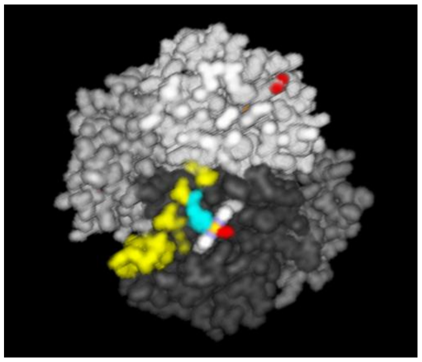
3.5 Contour spectrum



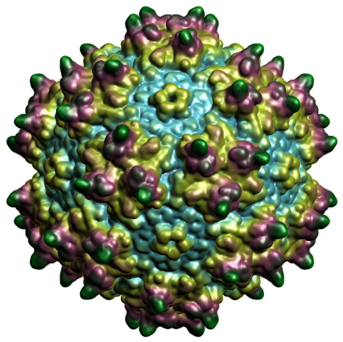
(a) Volume with colors assigned to secondary structures, blurred at the atomic level



(b) Volume with colors assigned to secondary structures, blurred at the residue level



(c) In order to show specific atoms like the iron in the heme structure of the hemoglobin, we can use a color map file.



(d) A virus (1lp3.pdb) rendered as an isosurface of a depth colored volume showing its surface features

Figure 3.2: Vector valued volumes constructed in different ways to bring out different features.

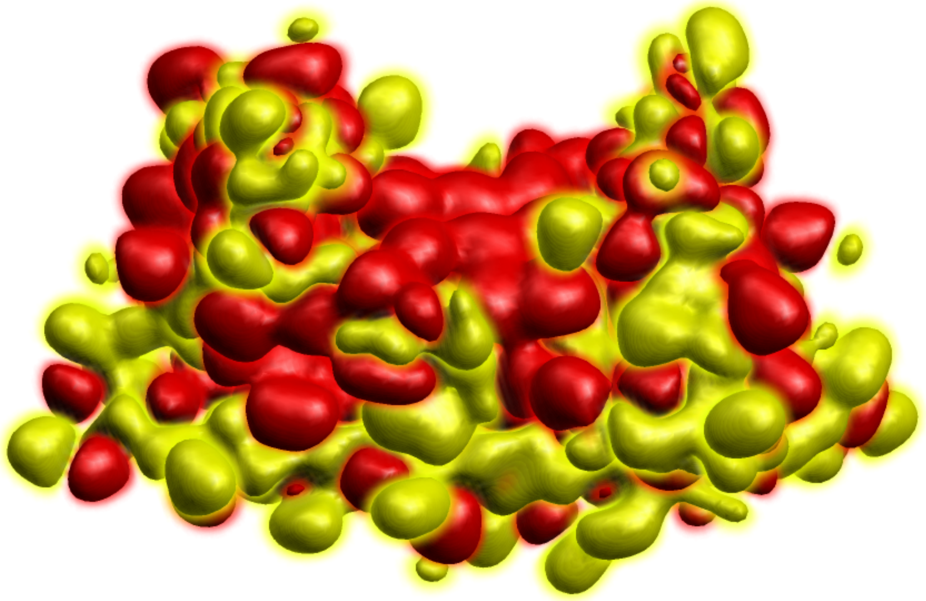


Figure 3.3: Hydrophobicity volume rendering of the dengue virus capsid protein (1r6r.pdb). The hydrophobic regions are in yellow and the hydrophillic regions are in red.

The contour spectrum is used for obtaining quantitative information about the volume files. The documentation for the contour spectrum can be found in the user manual for CCV's volume rover.

3.6 Pocket

Several of molecular features are bichemically significant as pockets are often active sites for ligand binding or enzymatic reactions, and tunnels are often solvent ion conductance zones. This pocket function extraction is also useful to compare protein structures based on molecular complementary space features. The pocket function gives a different way of computing similarity score and comparing proteins.

Using the graphical user interface, the users can compute the Pocket function of a molecule as shown in table 3.4

Table 3.4: Pocket function computation

- Load a molecule file if it is not already loaded.
- Select the molecule from the molecule browser.
- Select Utilities and Construct pockets from the menu bar.
- The generated volume and mesh are loaded in to memory.
- Highlight each file by selecting and check render checkbox to visualize, respectively
- Save each file with right mouse button

3.7 Curation

The selection of appropriate level sets for the quantitative visualization of three dimensional imaging or simulation data, is a problem that is both fundamental and essential. The selected level set needs to satisfy several topological and geometric constraints to be useful for subsequent quantitative processing and visualization. For an initial selection of an isosurface, guided by contour tree data structures, we detect the topological features by computing stable and unstable manifolds of the critical points of the distance function induced by the

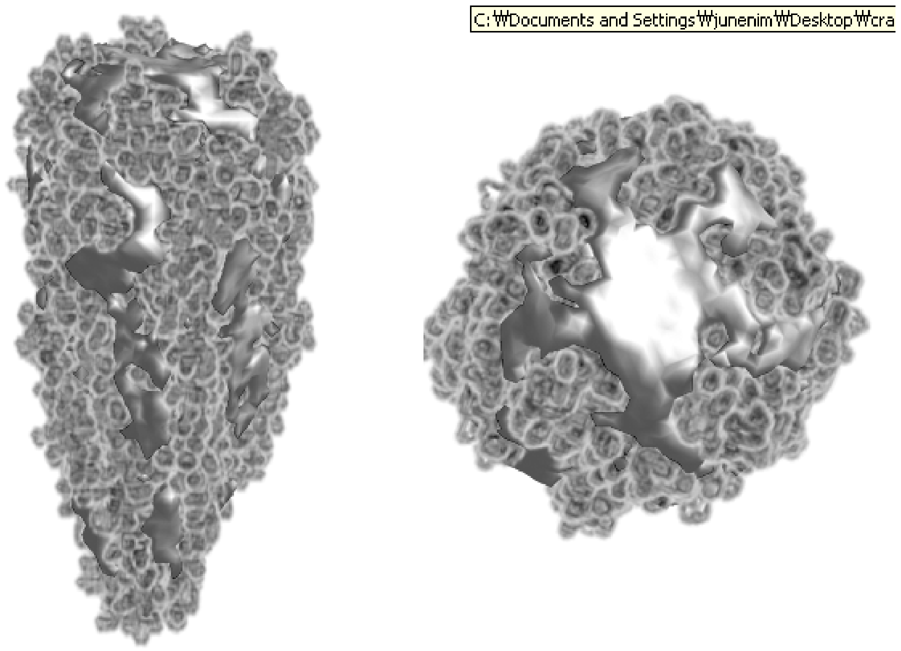
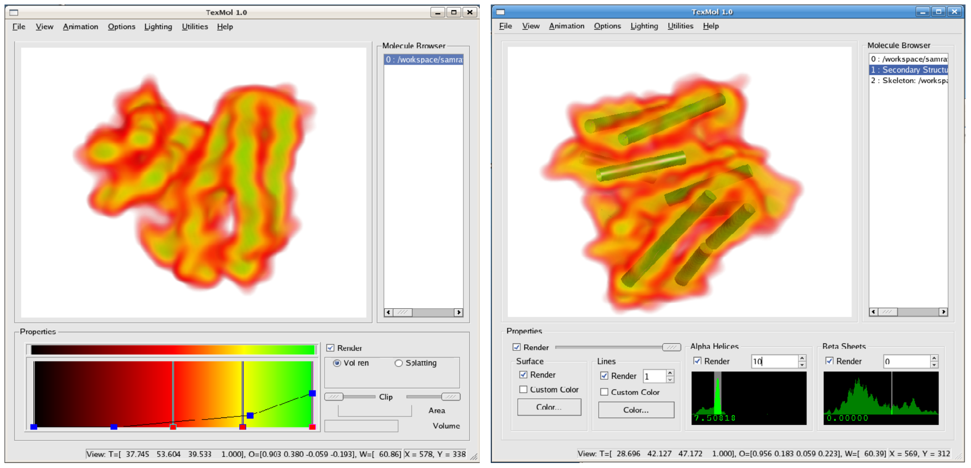


Figure 3.4: Combined isosurface and volume rendering of the Acetylicholine Receptor (2bg9.pdb). The pocket regions are identified with the shield (gray color).

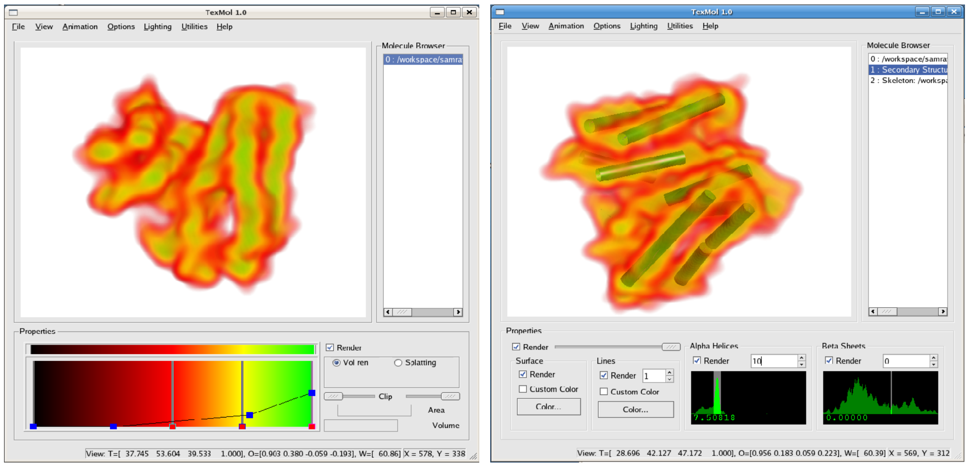


```
View: T=[  37.745   53.604   39.533    1.000], O=[0.903 0.380 -0.059 -0.193], W=[  60.86] X = 578, Y = 338
```

Lines: Render 1; Alpha Helices: Render 10; Beta Sheets: Render 0

```
View: T=[  28.696   42.127   47.172    1.000], O=[0.956 0.183 0.059 0.223], W=[  60.39] X = 569, Y = 312
```

Figure 3.5: Secondary Structure Elucidation from electron density volume data.



View: T=[37.745 53.604 39.533 1.000], O=[0.903 0.380 -0.059 -0.193], W=[60.86] X = 578, Y = 338

View: T=[28.696 42.127 47.172 1.000], O=[0.956 0.183 0.059 0.223], W=[60.39] X = 569, Y = 312

Figure 3.6: Secondary Structure Elucidation from electron density volume data.

isosurface. We further enhance the description of these features by associating geometric attributes with them. We then rank the attributed features and provide a handle to them for curation of the topological anomalies.

The steps for the curation is described in Table 3.5. A visual illustration of the process is given in 3.1.1.

3.8 Secondary Structure Elucidation from 3D Maps

Recent advances in three dimensional Electron Microscopy (3D EM) have given an opportunity to look at the structural building blocks of proteins (and nucleic acids) at varying resolutions. In TexMol, we have incorporated algorithm to detect the secondary structural motifs (α-helices and β-sheets) from proteins for which the volumetric maps are reconstructed at $5 - 10\mathring{A}$ resolution. The algorithm uses the tools from computational geometry and differential topology, specifically the computation of stable/unstable manifolds of certain critical points of the distance function induced by a suitably extracted molecular surface. Details of the theory and computation involved in various algorithmic steps are given in the accompanying paper [6].

The steps for the curation is described in Table 3.6. A visual illustration of the process is given in 3.1.1.

3.9 Meshing of Molecules

3.10 Protein-Protein Docking

Table 3.5: Curation

- Load a pdb file or an electron density map or a surface geometry.
- Select the molecule or map or geometry from the molecule browser.
- Select Utilities and COMPUTE POCKET-TUNNEL BY STABLE MANIFOLD from the menu bar.
- A pop-up menu asks for the number of pockets or tunnels that the user wants to compute. The user should enter two numbers if this information is already known. Otherwise any two large integers should capture all the pockets and tunnels that this molecular surface posseses.
- For input pdb, a molecular surface is extracted and in case of a volume representing the electron density map, a deafult isovalue is used to extract a molecular surface from the density map. If a molecular surface is given as input, no pre-processing is done.
- For the pre-processed (from pdb or volume) or the input geometry, the depressions (pockets) on the molecular surface, and the through holes (tunnels) are computed by the algorithm described in [5].
- The colored geometry files of the pockets and tunnels are loaded into the browser for visual inspection.
- Highlight each file by selecting and check render checkbox to visualize, respectively.
- Save each file with right mouse button

Table 3.6: Secondary Structure from EM Maps

- Load an electron density map ($.rawiv$) file.
- Use the volume rendering widget to extract an isosurface geometry.
- Select Utilities and SECONDARY STRUCTURE from the menu bar.
- This will compute the α-helices and the β-sheets of the molecule.
- After completion of the computation, a new panel will appear at the bottom which will expose the parameters related to the width of the helices and sheets. The overall distribution of the width of the skeletal structure of the molecule is drawn as a histogram. User can select a portion of the histogram and correspondingly the helices and sheets are displayed in the main rendering area.

Chapter 4

Scripting and animations

The computations defined in chapter 2.1.1 can also be performed in batch mode. This is useful when a large number of files need to be manipulated at once. Two other powerful features of *TeχMol* include

- Scripting.

- Animation.

In this chapter, we will go through simple examples for each of the above three features.

4.1 Batch-mode calculations

Electron density, hydrophobicity and curvatures can be estimated from the command line. The commands are as follows.

4.1.1 Electron density and hydrophobicity

The command line for creating an electron density or hydrophobicity map is:

TeχMol **-blur** [input molecule file name] [output volume file name] [dim1] [dim2] [dim3] [density type] [colored volume] [gaussian blobbiness] [color] [colormap file name] [gap]

The different parameters are:

[input molecule file name] The full path and name of the input molecule file.

[output volume file name] The full path and name of the output rawiv or rawv file.

[dim1, dim2, dim3] Dimensions of the output volume.

[density type] Enter 0 for electron density and 1 for hydrophobicity.

[colored volume] *true* if you want a rawv volume.

gaussian blobbiness We recommend a value of -2.3 to conform with previous research and bigger values, say -0.1, to get smoother shapes representing the volume at much lower resolutions.

color If the colored volume parameter was true, then we can either specify either color by structure or provide a color map file. If we do not provide a color map file, then this parameter is relevant. 0 stands for atom coloring, 1 for residue, 2 for secondary structure and 3 for chain coloring.

colormap file name The molecule can be colored according to some user defined colors, to perhaps highlight some structure which cannot be done in the more general method previous described. The file format for the *colormap file name* is given in the file format description page from CCV's website.

gap This should be used sparingly, to provide a gap of empty space around the requested dimensions. It is best left as 0.

4.1.2 Curvatures

There are two commands for generating curvatures. One is from a volume file with a given isovalue, where *TeχMol* extracts the mesh where we need to obtain the curvature, and second, where the user inputs the mesh directly. The two commands in order are:

TeχMol **-setcurvature** 1 input molecule file name output volume file name output mean curvature file name output gaussian curvature file name dim1 dim2 dim3 gaussian blobbiness isovalue number of grid subdivisions maximum function error

TeχMol **-setcurvature** 0 input molecule file name input surface file name output mean curvature file name output gaussian curvature file name dim1 dim2 dim3 gaussian blobbiness number of grid subdivisions maximum function error

The various parameters are described below.

0 and 1 differentiate between the two calls.

input molecule file name The full path and name of the input molecule file.

output volume file name The full path and name of the output rawiv or rawv file.

output mean curvature file name The full path and name of the output mean curvature file.

output gaussian curvature file name The full path and name of the output gaussian curvature file.

dim1, dim2, dim3 Dimensions of the output volume.

gaussian blobbiness We recommend a value of -2.3 to conform with previous research and bigger values, say -0.1, to get smoother shapes representing the volume at much lower resolutions.

number of grid subdivisions The higher this value, the more the precomputation and memory requirement. There is a cut off value around 10 to 20 where you get optimum speed for a given machine.

maximum function error The maximum allowable function error while calculating the curvatures.

Warning:While creating curvatures, if we overestimate the value of grid spacing, we could end up with very high memory usages and may have to kill the program.

4.2 Scripting

TexMol currently offers a limited but powerful set of functions as part of its scripting module. This module is available once *TexMol* is run in user interface mode. The parser command window can be opened from Utilities - script from the main menu bar. Here users can enter commands, select one or more and execute them. Some of the commands available are listed below.

bool blur(int argc, char* argv[]);
bool setCurvature(int argc, char* argv[]);
bool outGridPositions(int argc, char* argv[]);
bool classifyPoints(int argc, char* argv[]);
bool growOut(int argc, char* argv[]);
bool getSurface(int argc, char* argv[]);
bool evolve(int argc, char* argv[]);
bool writePDB(int argc, char* argv[]);
bool writeGOA(int argc, char* argv[]);
bool depthColor(int argc, char* argv[]);
bool printPDBInformation(int argc, char* argv[]);
bool getMaxDistanceFromPoint(int argc, char* argv[]);
bool addNewDataSet(int argc, char* argv[], MoleculeVizMainWindow *mWindow);
bool splitView(int argc, char* argv[], MoleculeVizMainWindow *mWindow);
bool deleteData(int argc, char* argv[], MoleculeVizMainWindow *mWindow);
bool deletePrevData(int argc, char* argv[], MoleculeVizMainWindow *mWindow);
bool deleteAllData(int argc, char* argv[], MoleculeVizMainWindow *mWindow);
bool setVisible(int argc, char* argv[], MoleculeVizMainWindow *mWindow);
bool setVisiblePrev(int argc, char* argv[], MoleculeVizMainWindow *mWindow);

bool saveImage(int argc, char* argv[], MoleculeVizMainWindow *mWindow);
bool setGridVisible(int argc, char* argv[], MoleculeVizMainWindow *mWindow);

4.3 Animations

This module is mainly useful for making movies. Animations can be saved as a sequence of images in *TeχMol* . We allow the user to load a data set, render it and transform the view point, and save trajectories and play it back. There are actually two steps in the animation process.

1. Create and save a trajectory.
2. Save the animation based on a trajectory.

4.3.1 Creating a trajectory

Data sets can be quite large, making them hard to interact with. Hence, it is sometimes useful to save a trajectory using a low resolution data set and later use it to render a large data set in high resolution. In order to create the trajectory it is useful to have a low resolution data set which the user can comfortably rotate, zoom and translate interactively. To save a trajectory, follow the procedure outlined in table 4.1

4.3.2 Saving animations

Before you can save a sequence of images, you need to create a trajectory as described in section 4.3.1.

⍂	Warning:Disable the screensaver as saving images in high resolution can take a long time. We also interpolate between mouse movements using linear interpolation to get better smoothness, causing the movie to be longer than expected.

Table 4.1: Creating and saving a trajectory

- Load and display the data set or set of data sets you want to create a movie of.
- Use the mouse to move them into the correct starting viewing position.
- Click Animation, Start recording from the menu bar.
- Enter and save the file name of the trajectory file. It does not require any specific file extension.
- On pressing OK, the recording is on. Any mouse movement in the rendering area is saved in to the file, along with the time.
- Once you are done with transforming the data, click Animation, Stop recording in the menu bar.
- You can see if the trajectory was satisfactory by playing it back by clicking on Animation, Playback animation from the menu bar.

Table 4.2: Saving an animation

- Load the high resolution data sets to save.
- keep the screen resolution and window size as high as required.
- Open the trajectory file by selecting Animation, Record animation from the menu bar.
- *TeχMol* will automatically begin to replay the trajectory and save images.

Acknowledgements

TeχMol was developed at the Computational Visualization Center (CVC) under the direction of Prof. Chandrajit Bajaj. The constributors are

- Albert Chen
- Rezaul Chowdhury
- Peter Djeu
- Samrat Goswami
- Bongjune Kwon
- Alex Lee
- Vinay Siddavanahalli
- Matt Strange
- Anthony Thane
- John Wiggins
- Xiaoyu Zhang
- Wenqi Zhao

Grants

Bibliography

[1] Chandrajit Bajaj, Peter Djeu, Vinay Siddavanahalli and Anthony Thane *$Te\chi Mol$* : Interactive Visual Exploration of Large Flexible Multi-component Molecular Complexes IEEE Visualization 2004, pp. 243-250.

[2] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne: The Protein Data Bank. Nucleic Acids Research, 28 pp. 235-242 (2000).

[3] Todd J. Dolinsky, Jens E. Nielsen, J. Andrew McCammon, and Nathan A. Baker: PDB2PQR: an automated pipeline for the setup of PoissonBoltzmann electrostatics calculations. Nucleic Acids Res. 2004 July 1; 32

[4] Chandrajit Bajaj and Vinay Siddavanahalli, Fast Error-bounded Surfaces and Derivatives Computation for Volumetric Particle Data, ICES Report 06-03, January 2006.

[5] Chandrajit Bajaj and Andrew Gillette and Samrat Goswami, Topology Based Selection and Curation of Level Sets. To appear as a book chapter of Mathematics and Visualization.

[6] Chandrajit Bajaj and Samrat Goswami, Automatic Fold and Structural Motif Elucidation from 3D EM Maps of Macromolecules. ICVGIP 2006.

[7] X. Zhang and C. Bajaj, Extraction, Visualization and Quantification of Protein Pockets. *6th Ann. Int. Conf. on Comput. Sys. Bioinfo. CSB2007.*

[8] W. Zhao, G. Xu and C. Bajaj, An algebraic spline model of molecular surfaces. *Proc. ACM Solid and Physical Modeling*, pp. 297-302, 2007.

[9] C. Bajaj, G. Xu and X. Zhang, Smooth surface constructions via a higher-order level-set method. *Pro. CAD/GRAPHICS 2007*

[10] X. Zhang and C. Bajaj and B. Kwon and T. Dolinsky and J. Nielsen and N. Baker, Application of New Multiresolution Methods for the Comparison of Biomolecular Electrostatic Properties in the Absence of Structural Similarity. *Multiscale Modeling and Simulation*, **5** (4), pp. 1196-213, 2006.

[11] S. Goswami, A. Gillette and C. Bajaj, Efficient Delaunay Mesh Generation From Sampled Scalar Functions. *Accepted in International Meshing Roundtable 2007*.

[12] C. Bajaj, J. Castrillon-Candas, V. Siddavanahalli and Z. Xu, Compressed Representations of Macromolecular Structures and Properties. *Structure*, **13** (3), pp. 463-471, 2005.

4.4 License Agreement

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

1. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder (THE COMPUTATIONAL VISUALIZATION CENTER at THE UNIVERSITY OF TEXAS AT AUSTIN) or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

   A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

   The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

   "Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

2. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 2 above, provided that you also meet all of these conditions:

   - The modified work must itself be a software library.
   - You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
   - You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
   - If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful. (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 3d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

CREDITS

WE REQUEST THAT YOU AGREE TO ACKNOWLEDGE THE USE OF THE SOFTWARE THAT RESULTS IN ANY PUBLISHED WORK, INCLUDING SCIENTIFIC PAPERS, FILMS AND VIDEOTAPES BY CITING THE REFERENCES IN CODE FILE AND DOCUMENTATION BOUNDED WITH THE SOFTWARE.

C. Bajaj, P. Djeu, V. Siddavanahalli and A. Thane
TeχMol : Interactive Visual Exploration of Large Flexible Multi-component Molecular Complexes, *Proc. IEEE Visualization 2004*, pp. 243–250.

This software has been developed at the Computational Visualization Center at The University of Texas at Austin under

Dr Chandrajit Bajaj
Computational Applied Mathematics Chair in Visualization
Professor of Computer Sciences
Director of Computational Visualization Center
The Institute of Computational Engineering and Sciences
The University of Texas at Austin