Labeling a Molecular Triangle Meshes

Cody Robson

Abstract

This project addresses the problem of molecular visualization and the application of decals or labels onto a molecule mesh. Often biochemists will want to mark a region of interest or label a specific protein residue on a molecule in real time. This project looks into the application of an existing technique for applying textures to an arbitrary surface without texture coordinate information. Then, a system is built to apply markers or manually create and orient text strings on the mesh surface and begin to address the problem of automatically labeling an entire molecule.

Goals

The ultimate goal of this approach is to allow the labeling of arbitrary regions on a molecule mesh while maintaining all the visual queues that convey the actual shape of the molecule at and around the region. Also, solutions that do not specifically color a region to differentiate it from another would be favored, as biochemists often want to use color to display actual physical properties of the molecule a given mesh is representing, such as molecular charge, specific atoms, etc. Given these goals, a billboard-based approach would not be a viable solution, as it would obstruct the molecular surface beneath each billboard, and the automatic alignment of billboards as the molecule is rotated would be disorienting to users.

Existing Techniques

There exists a number of techniques for labeling or creating texture coordinates for arbitrary meshes. [Schmidt06] was chosen because it is both quick and does a good job of preserving lengths as a 2D texture is wrapped around a 3D surface, which is especially important for font rendering if it is to remain legible. Other options would be displaying text as 3D geometry protruding from the surface itself [Slabaugh05] but since a goal of this project is to retain the visual queues of the underlying molecular shape, adjusting the mesh geometry

Figure 1: Applying an image to the mesh surface.

would at best occlude the actual shape of the molecule and at worst provide false visual queues.

Applying Labels Manually

In the most basic implementation, a user would be able to create labels before hand as image files and apply them to the mesh. The user input required for this process was the image file, a size for the texture and an orientation. The user can interactively place a tangent plane and adjust its size and orientation before the image is applied. The process of mapping the 2D texture onto the 3D surface is shown in reverse, as the vertices on the underlying 3D mesh are drawn on the tangent plane.

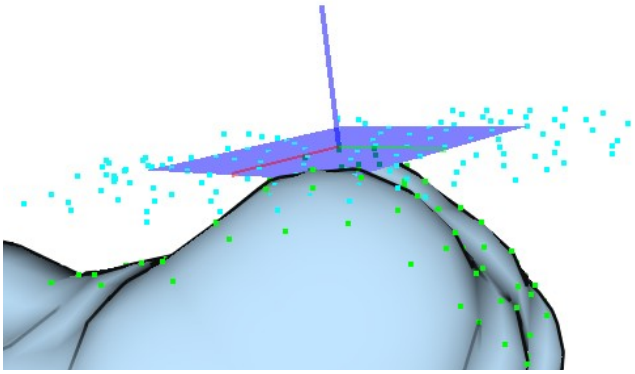

Figure 2: Orienting the tangent plane and viewing the projected vertices. Green vertices are on the actual mesh below, and blue vertices are their projected positions in the tangent plane.

Creating Text Labels

Because creating labels consisting of just text in image editing programs is tedious, a system for creating small text labels was integrated into the program. Now, after the user has sized and oriented their tangent plane on the mesh, a canvas is opened in which they are allowed to position a text string. Because some labels would benefit from being curved, the text follows a cubic b-spline with an arbitrary number of knots (at least four) that the user is allowed to position as well as adjust text size and kerning values. For the sake of kerning simplicity, a fixed-sized font was used, but of course this could be extended to a more complex font renderer. Each letter in the string is rendered to its own square texture so that it can be scaled and oriented along the spline, and the resulting image is then rendered to a larger texture to be applied through the discrete exponential map system onto the mesh. Because the typical size of labels is small in comparison to the mesh, lossy OpenGL texture compression was used to cut down on memory usage at little to no loss in visual quality.

Labeling Regions of Interest

Protein molecules are divided into smaller, intertwined collections of atoms called residues. Given the information of which vertices belong to which residue region, the labeling process can be adapted to best differentiate these residues from one another. Simply giving a residue a unique color and applying that to the mesh is a trivial solution, but as previously stated the use of color is

not ideal as its best saved for physical properties of the molecue.

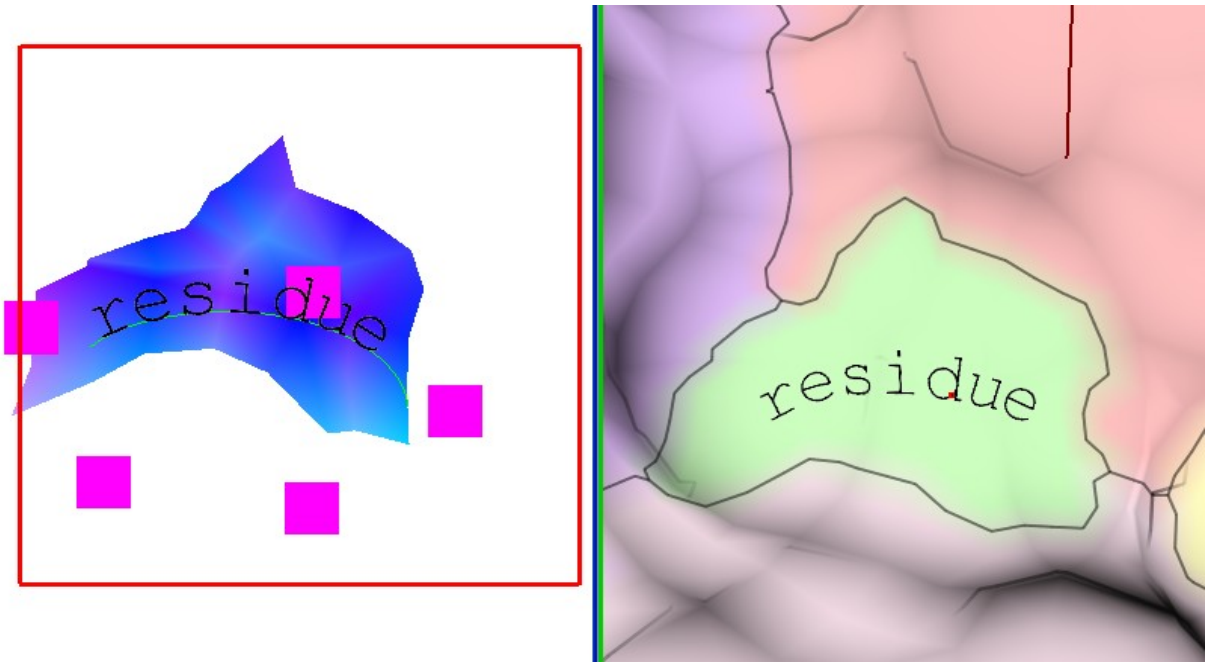

Figure 3: Left: Manually creating a text label along a b-spline with the residue normal data in the background. Solid blue indicates completely flat areas, whereas cyan and magenta indicate deviations in the normal in the U and V directions, respectively. The magenta squares are the knots defining the bspline, drawn in green. Right: The resulting text label.

As an extension to the text label creator, the vertices involved in the residue (or any vertex grouping deemed a region of interest) are painted on the background of the canvas to show the user exactly where the text will be placed in the region. In the most simple case, a border detection algorithm can produce and display the border of the residue on the canvas so that the user can place a label neatly in the interior of the region. Three dimensional data like vertex normals, curvature, or ambient occlusion (a shading prevalent in molecule rendering) can be displayed on the canvas to inform the user more details of the region beyond its 2D projected shape.

Because of the difficulties of unwrapping a very non-developable 3D surface onto a 2D plane (or vice versa) without dividing it (mesh cutting techniques are beyond the scope of this project), sometimes mesh triangle faces overlap in the 2D texture space, causing glaring artifacts for any label that ventures into that part of the region. Without undergoing to task of remeshing or optimizing the texture coordinates to minimize this problem, these areas are easily identifiable and also can be shown to the user in the canvas so at the very least they can avoid placing text in the offending region. Note that this only happens regularly on very large residue regions that wrap around hills and valleys in the mesh, so there is almost always plenty of flat, correctly projected areas in a given residue for text labeling.

The Beginning of Automating the Process

Once the process of labeling a protein residue, or arbitrary region of interest on a triangle mesh, can be done manually, automation is the next logical extension. For a protein molecule, every atom is a part of some residue, so on the extrapolated triangle mesh, every vertex belongs to one and only one residue region. Using these vertex groups as the residue regions would leave strips of unlabeled faces along the borders of the residues. To make regions a per-face definition, a greedy residue growing algorithm was implemented that just identifies faces with vertices belonging to more than one residue region and adds all three vertices (sans duplicates) to one of the face's possibly three residues. More sophisticated processes like dividing these border triangles could be implemented, but seeing how the positions of these vertices are completely up to the density of the surface-constructing algorithm that generated the mesh from a protein data file, its easier to subdivide as needed and be greedy about the growing process.

Some of these regions of interest may be disconnected, especially with protein residues that in reality are connected in the interior of the molecule but such information is lost in the process of surface generation. A simple depth or breadth first search in each of these regions will produce a set of connected regions that more easily lend themselves to the discrete exponential map 2D projection process.

For each of these regions, a starting point for the discrete exponential map projection process is needed. The best observed solution is to rate each vertex by summing one minus the dot product of its normal versus the normal of every other vertex in its region, resulting in a score of 0 for each perfectly aligned normal in its group, 1 for an orthogonal normal, and 2 for an opposite normal. The vertex with the lowest score has a normal that best 'agrees' with its region and is a suitable candidate for being the starting point for the 2D projection process. Other methods were tested, including scoring the distance from each vertex's tangent plane to each other vertex, or weighting one of the above methods by euclidean distance to each other vertex, but the results were either worse

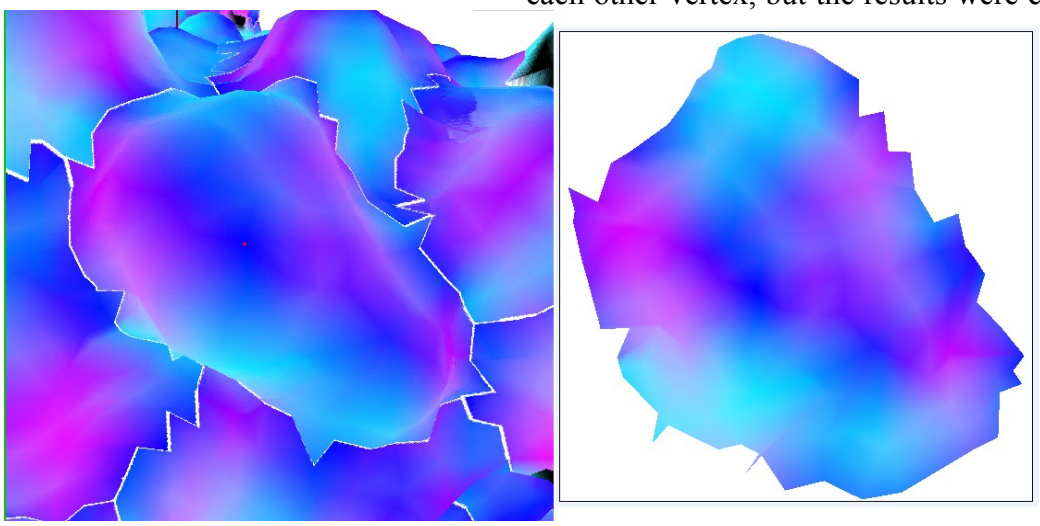

Figure 4: Left: A residue on the mesh with the normal data image applied to it. Right: The unwrapped region in 2D.

or no better at preventing the overlapping regions described earlier.

At this point, every face is assigned to a group, and every group has a starting vertex for the projection into 2D texture coordinate space, but it would be nice if these regions were oriented so that there was some coherence from one region to the next, especially if text is to be placed on each of them and expected to be readable from a single viewpoint. This is an example of the 'hairy ball' problem, and without going into a complicated optimization process, a few quick, greedy methods were tried and all faired about the same. One approach was to just orient each tangent plane's (the 2D texture space oriented at the starting vertex) V axis with the world up vector. This led to regions with similar orientation around the 'equator' of the mesh, but vastly disagreeing regions in any area of the mesh facing upwards or downwards. A simple addition to this greedy algorithm was to give each region a 'confidence' level that scaled with how orthogonal a starting vertex's normal is to the world up vector. The logic here is that a normal that is perfectly orthogonal to the world up vector can align its tangent plane's V axis with the world up vector perfectly (and thus be most confident), and a normal that is aligned with the world up could only make its tangent plane's V access orthogonal to the world up vector (and thus be least confident). Each region then could blend its initial orientation (from the previous method) with its most confident neighbor's initial orientation. Because the upward and downward facing regions would still disagree so much, perhaps the alignment of the world up vector is simply arbitrary, and an approach could be just to orient a single region by any method, and to greedily grow outward orienting its neighbor to most closely match that region until all residues are oriented. None of these solutions are optimal but they are simply meant to be better than nothing. Note that because the world up vector is used, the mesh would have to be relabeled at the will of the user if they wish to change the molecule's orientation significantly.

Automatic Labeling

From here, each of these regions can be labeled with a simple alpha-blended stripe texture so that existing mesh coloring remains intact but each residue region can be visually differentiated from each other, but not outright identified. The more

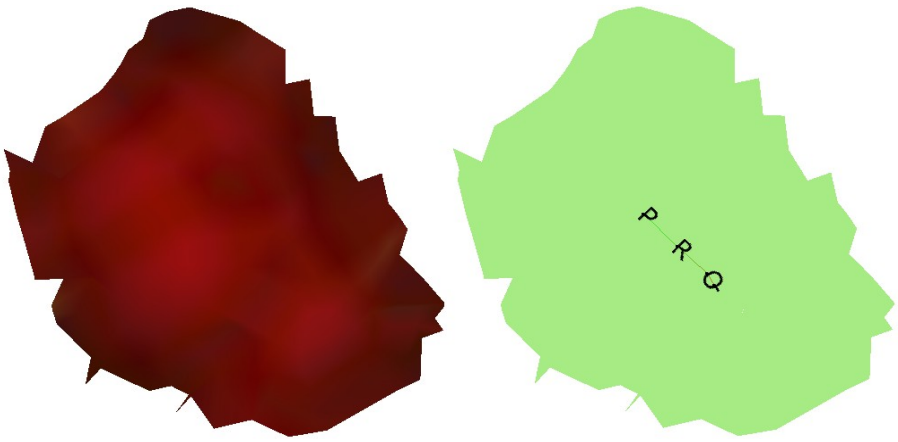

Figure 5: Left: The ambient occlusion data image, darker areas indicate low visibility. Right: Initial automatic labeling results.

involved process of labeling each of these regions with its residue name and identification number (as there may be more than one instance of a type of residue) can be built by defining existing cartography principals of map area labeling as a optimization problem. This project did not venture into that process but made available as much information as possible to aid in that project. For each residue region, a list of vertices involved (both in 3D mesh space and 2D texture space) and several data images are written out to be used in the automatic text labeling problem. Each of these data images were produced earlier on the text decal creation canvas to give the user a visual reference. Now, its the actual numerical per-pixel data of these images that are relevant.

Once the whole process is run, one of these data images are applied to each region as if it were the final text label. An initial version of the automatic text label generation system was developed by Gregory Cipriano and Professor Michael Gleicher based on principals of cartographic map labeling, namely using the medial axis of a given area to find the 'spine' of a region and use that as a starting point for the text placement. [Dorschlag03] An optimization process involving the data images and this medial axis is used in the initial versions of this program.

A label swapping process was implemented to easily switch these initial data image labels to the final outputted labels from that program. Additionally, label save and load functionality was added by simply saving lists of decals, and for each decal lists of vertices and their texture coordinates so that the computation need not be run again.

Test Region Creation

To test these methods on non molecular meshes, and they tend to be quite large, a process of building residue groups from mesh vertex colors was added. Because programs like Blender use 'paint brush' like tools to color vertices, it can be hard to paint vertices in a specific group all exactly the same color. Therefore, the program assigns a vertex to the closest one of eight color regions (white, black, red, blue, green, yellow, cyan, and magenta). Because the code to identify disconnected residues was already in place, an arbitrary number of test regions can be created this way so long as test regions of the same color are disconnected. This means the user must only be able to 8-color a mesh, which should be trivial. Additionally, since vertices between two regions may get a blend of the two colors that is closer to neither of them (like in the case of a yellow and blue region producing a white vertex), a simple voting system for each vertex considers its 1 or 2 jump neighbors to ensure that vertices on borders of two regions will fall into one or the other, and not create a third, unwanted region.

Conclusion

The novel contribution of this project was the

application of texture decaling to apply text labels to a protein surface, and provide the infrastructure necessary to build a automatic text labeling system. Most of the work was, admittedly, a reimplementation of existing work, but the problem of protein molecule labeling has gone largely un-addressed to my knowledge.

References

[Schmidt06] Schmidt, Ryan; Grimm, Cindy; Wyvill, Brian. *Interactive Decal Compositing with Discrete Exponential Maps*, ACM Transactions on Graphics Vol 25 Issue 3 July 2006.

[Slabaugh05] Slabaugh, G.; Mihalef, V.; Unal, G.. *A Contour-Based Approach to 3D Text Labeling on Triangulated Surfaces,* 3DIM 2005 Vol 13 Issue 16 June 2005.

[Dorschlag03] Dorschlag, D.; Petzold, I.; Plomer, L.. *Placing Objects Automatically In Areas Of Maps*, University of Bonn Institute for Cargotraphy and Geoinformation. Hosted at http://www.ikg.uni-bonn.de/

petzold/labplace/publications/

doerschlag_petzold_icc03.pdf