Progress on STEP Visualization
NETL/ WVU NARA project

Tim McGraw

September 2008

1 Introduction

We are currently developing techniques for visualizing collections of STEP files. Here we will describe the methods used and present preliminary results. Our preliminary results use small synthetic graphs as input to the system. In these examples the nodes may represent STEP files, and the edges represent references between them, just as the assembly of a large structure may reference smaller subassemblies. At another scale the graph may represent the structure within a single step file. It is important to note that the nodes and edges we visualize are not necessarily the vertices and edges of the 2D/3D geometry contained in a STEP file. We are developing large graph visualization methods as an approach to solving the problem of visualizing STEP data and the relationships among datasets in a repository.

The traditional approach to displaying graphs is to render a small symbol (a circle, or sphere) for each node, and then draw the edges in the graph as curves between the nodes. Such a visualization becomes unsatisfactory for moderately sized graphs. The overlapping edges leads to visual clutter which can obfuscate the relations between nodes. We illustrate part of our approach with an example. Consider the two graphs in Figure 1. Are they equivalent? (The graphs are equivalent if there is a one-to-one mapping between the nodes in one graph and the nodes in the other graph such that the mapping preserves edge connectivity). Even for this small example it is not trivial to determine the equivalence visually. Now consider the two shapes in Figure 2. Are they equivalent? In this case they are equivalent, and it is fairly easy to come to this conclusion by visual inspection. The intuitive and visual notion of shape is invariant to many transformations. For instance, one can easily identify a cube, regardless of position, orientation and size. There is another important connection between the graph in Figure 1 and the shape in Figure 2. The given graph is not embeddable in the plane - there is no set of node positions in the plane such that no two graph edges overlap. The given graph is, however, embeddable on the torus as shown in Figures 6 and 7. Our visualization approach is to map graph structure to geometric shape and other visual properties which are easily discernable. We

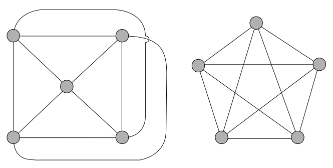


Figure 1: Two graphs. Are they the same? It is difficult to determine visually.

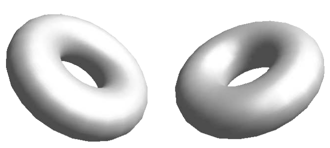


Figure 2: Two shapes. Are they the same? It is easy to determine visually.

consider the torus in this case to be a visual representation of the graph. In addition, we would also like to support the option of seeing the nodes and edges embedded on the surface. The physically-based method for achieving this is described in section 2.1.

The graph in Figure 1 is known as K_5, the completely connected graph of 5 nodes. It can be proven that every graph which is not embeddable in the plane has K_5 or $B_{3,3}$ (the complete bipartite graph with 3 nodes in each partition) as a subgraph. Furthermore, by removing the K_5 and B_{3-3} subgraphs from a nonplanar graph we can obtain a graph which is embeddable in the plane.

The approach we will take for visualizing global graph structure is to represent the graph as a surface which that graph is embeddable on. In general, a graph may have multiple K_5 and $B_{3,3}$ subgraphs, so we may need a high genus surface on which to embed the graph. The genus of this surface can be determined by graph segmentation since the number of K_5 and $B_{3,3}$ subgraphs determine the genus of the visualization surface.

The embedding surface shape alone is not the sole method of visualizing a graph. However, this surface acts as a canvas for visualizing other properties of the graph. We propose a hierarchical mapping from graph structure to visual appearance for further visualization which is summarized in Figure 3. We also propose to take a multiscale approach to visualization. Using clustering approaches we can collapse highly connected groups of nodes at one scale to a single node at the next higher scale. A spectral clustering approach is described in section **??** which will allow us to deal with a moderate number of nodes at each scale during visualization will and require render embedding surfaces of moderate genus. Strictly speaking, the full graph is then visualized as

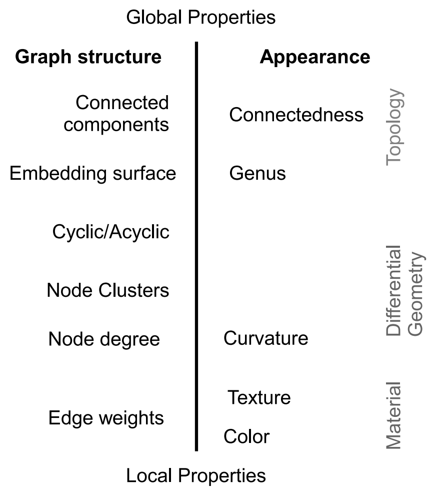


Figure 3: A hierarchy of graph properties and visual properties.

a sequence of surfaces with increasing scale.

2 Methods

2.1 Graph Drawing Algorithm

Given an arbitrary graph and a manifold, consider embedding the graph on the manifold. In order to make the graph visually comprehensible we wish to embed all the vertices and edges of the graph on the manifold so that there are no overlapping edges. Our approach is to convert the given graph into a mass-spring system [1]. In order to constrain the motion of the vertices within the manifold we utilize constrained dynamics [2]. At the same time, we apply repulsive electric forces and damping forces among all the vertices so that they maintain a certain distance from each other. We initialize the positions of the particles randomly or manually and simulate the motion of the particles until those positions converge to their steady state.

2.1.1 Mass-spring System Representation of Graph

Consider a graph $G = (V, E)$ where V is a set of N vertices $\{v_1, v_2, \ldots, v_N\}$ and E is a set of M edges $\{e_1, e_2, \ldots, e_M\}$. Each edge is given by a pair of vertices (v_i, v_j), $i, j = 1, \ldots, N$.

We construct a particle system that corresponds to G as follows:

- For each vertex, assign a particle.
- For each edge, subdivide the edge into multiple segments and assign a spring to each segment and a particle to each joint between the setments. Let the total number of the particles that are assigned to the joints be L.

Thus in total we have $K = N + L$ particles in the particle system. Let the total number of the springs be K_s. The motion of this particle system is governed by Newton's second law of motion

$$\mathbf{M}\ddot{\boldsymbol{q}} = \boldsymbol{Q} + \hat{\boldsymbol{Q}} \tag{1}$$

where $\boldsymbol{q}$ is the vector of the position of particles defined as

$$\boldsymbol{q} = [x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_K, y_K, z_K]^T,$$

$\mathbf{M} = \mathrm{diag}(m_1, m_1, m_1, m_2, m_2, m_2, \ldots, m_K, m_K, m_K)$ is the mass matrix, $\boldsymbol{Q}$ is the external force, and $\hat{\boldsymbol{Q}}$ is the constraint force.

The external force $\boldsymbol{Q}$ is a sum of the repulsive electric forces $\boldsymbol{F}^{(e)}$, spring forces $\boldsymbol{F}^{(s)}$, and damping forces $\boldsymbol{F}^{(d)}$ given by

$$\boldsymbol{Q} = \boldsymbol{F}^{(e)} + \boldsymbol{F}^{(s)} + \boldsymbol{F}^{(d)}. \tag{2}$$

Each term in the right hand side of Equation (2) are comprised of the combination of the following components. The repulsive electric force between the particle p_i and p_j follows Coulomb's law given by

$$\boldsymbol{f}_{ij}^{(e)} = k_e \frac{q_i q_j}{||\boldsymbol{r}_{ij}||^2} \frac{\boldsymbol{r}_{ij}}{||\boldsymbol{r}_{ij}||}$$

where k_e is a coefficient that determines the strength of the force, q_i is the charge of the particle p_i, and $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$ where $\boldsymbol{r}_i$ is the position vector of the particle p_i. The spring force between the particle p_i and p_j follows Hooke's law given by

$$\boldsymbol{f}_{ij}^{(s)} = -k_s(||\boldsymbol{r}_{ij}|| - d_{ij}) \frac{\boldsymbol{r}_{ij}}{||\boldsymbol{r}_{ij}||}$$

where k_s is the spring constant and d_{ij} is the rest length of the spring. Finally the damping force for each particle p_i is given by

$$\boldsymbol{f}_i^{(d)} = -k_d \boldsymbol{v}_i,$$

where $\boldsymbol{v}_i$ is the velocity of particle p_i.

Thus we have computed the external force $\boldsymbol{Q}$.

2.1.2 Constrained Dynamics

In order to solve Equation (1), we need to know the constraint force $\hat{\boldsymbol{Q}}$. This is the force that constrains the motion of the particles on the given manifold. The manifold is given as an implicit surface $F(x, y, z) = 0$. We wish all the particles' motion to be constrained on this surface. Thus, $F(q_{3i}, q_{3i+1}, q_{3i+2}) = 0$, $i = 1, \ldots, N$. We combine all these constraint equations into a vector equation $\boldsymbol{C}(\boldsymbol{q}) = \mathbf{0}$. To calculate $\hat{\boldsymbol{Q}}$ we utilize constrained dynamics [2]. Utilizing the formulation described in [2] the final equation to be solved can be described as a linear system

$$\mathbf{JWJ}^T\boldsymbol{\lambda} = -\dot{\mathbf{J}}\dot{\boldsymbol{q}} - \mathbf{JWQ} - k_s\boldsymbol{C} - k_d\dot{\boldsymbol{C}} \tag{3}$$

where $\mathbf{W} = \mathbf{M}^{-1}$, $\mathbf{J} = \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{q}}$ is the Jacobian, $\boldsymbol{\lambda}$ is the Lagrange multiplier, k_s and k_d are feedback parameters for numerical stabilization. The constraint force $\hat{\boldsymbol{Q}}$ is given by

$$\hat{\boldsymbol{Q}} = \mathbf{J}^T\boldsymbol{\lambda}. \tag{4}$$

We solve Equation (3) for $\boldsymbol{\lambda}$ and then constraint force $\hat{\boldsymbol{Q}}$ is obtained by Equation (4). Now that we know all the terms in the right hand side of Equation (1), we can solve it for $\boldsymbol{q}$. Eventually the value of $\boldsymbol{q}$ will converge to its steady state (due to the damping force) and thus the particles' positions are determined.

2.2 Rendering Manifolds

Presently, the embedding surface is rendered with a simple shading model. Future work will involve assigning material properties and local perturbations which will represent graph structural information. We use the marching cubes algorithm [3][4] for rendering the manifold since it is represented an implicit surface.

2.3 Hyperbolic Geometry

Hyperbolic geometry creates a fisheye lens effect. This characteristics is useful to display and navigate large graphs in a limited display area. There are various methods that implement the hyperbolic geometry [5][6][7]. We used a simple geometric method shown in Figure 4 to examine the effect of the hyperbolic geometry. The transformation is done by the following steps:

1. The input coordinates $P(p_x, p_y, 0)$ are transformed by the hyperboloid

$$H : \left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 - \left(\frac{z}{c}\right)^2 = -1.$$

2. Project $P(p_x, p_y, 0)$ onto H and let the projection be $P'(p_x, p_y, p_z)$.

3. Calculate the intersection $Q'(q_x, q_y, c)$ of line OP$'$ and the plane $z = c$.

4. Project Q' onto $z = 0$ and let the projection be $Q(q_x, q_y, 0)$.

In future work this type of mapping will be useful for navigating and visualizing the planar portions of the graph.

2.4 Spectral Clustering

Clustering is an important part of our proposed multiscale visualization technique. It will be used to minimize visual clutter at each scale of the visualization. Spectral Clustering [8][9][10] is a clustering method that has many fundamental advantages compared to traditional clustering methods. One of the advantages is that it can cluster data points that are not necessarily comprised of convex subsets. We apply this method to find distinct clusters in the graph. We examined the unnormalized spectral clustering algorithm that is described in [8]. The algorithm is as follows:

- Input: Similarity matrix $S \in \mathbf{R}^{n\times n}$, number k of clusters to construct.
- Construct a similarity graph. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- Compute the first k eigenvectors $u_1, \dots, u_k$ of L.
- Let $U \in \mathbf{R}^{n\times k}$ be the matrix containing the vectors $u_1, \dots, u_k$ as columns.
- For $i = 1, \dots, n$, let $y_i \in \mathbf{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\dots,n}$ in $\mathbf{R}^k$ with k-means algorithm into clusters $C_1, \dots, C_k$.
- Output: Clusters $A_1, \dots, A_k$ with $A_i = \{j | y_j \in C_i\}$.

3 Preliminary Results

The methods of the previous sections have been implemented and tested on small synthetic graphs.

3.1 Graph Embedding on Torus

We attempted to embed a perfect graph with five nodes (K_5) on a torus. K_5 is known to be impossible to embed on a plane and it requires a torus with genus one to be embedded on. The equation of the torus that we used is given by

$$F(x, y, z) = (x^2 + y^2 + z^2 + r_1^2 - r_2^2)^2 - 4r_1^2(x^2 + y^2) = 0$$

where r_1 is the major radius and r_2 is the minor radius. We manually initialized the position of particles and edges as in Figure 6. Starting with this initial condition the positions of particles evolve until they reach the steady state. Figure 7 shows the final (converged) positions of the particles. We confirmed that our algorithm creates reasonably smooth graph layouts.

3.2 Hyperbolic Geometry

We implemented the hyperbolic mapping technique and generated the images of large graphs. An example output is shown in Figure 5. Note that it creates the fisheye lens effect that emphasizes the detail in the center. Though the detail in the periphery is greatly distorted, some features are still discernable. By allowing the user to change the location of the center of projection it is possible to implement a movable lens feature in the user interface of our proposed software. This will be useful for navigating and visualizing the large planar portions of graphs. Adopting this method to the nonplanar regions is proposed for future work.

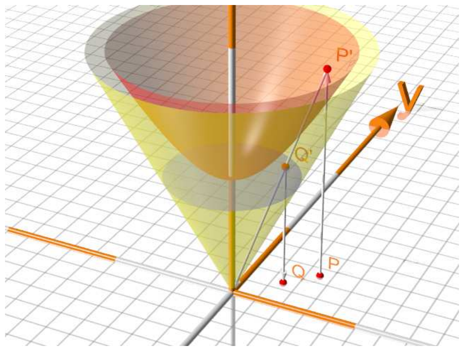


Figure 4: Hyperbolic geometry.

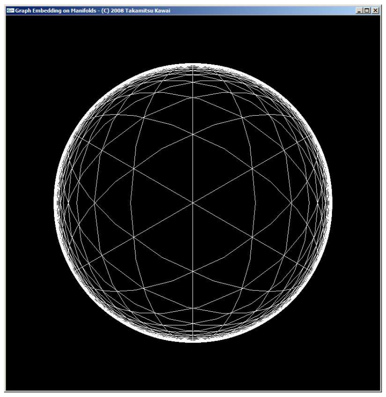
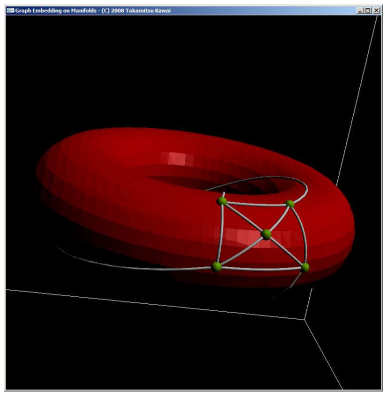


Figure 5: Effect of hyperbolic geometry.



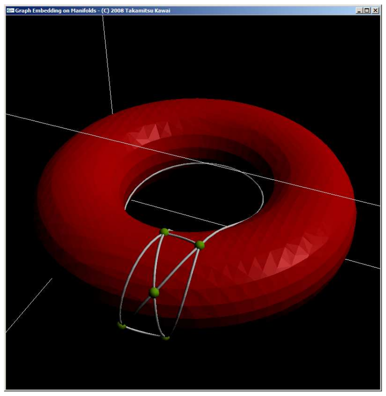


Figure 6: Graph on a torus (initialized). Figure 7: Graph on a torus (converged).

3.3 Spectral Clustering

We implemented the spectral clustering technique **??** and examined the performance with synthetic inputs. We show the results for 50 data points that are expected to be clustered into 3 clusters in Figure 8. The similarity graph is constructed by the Gaussian similarity function $s(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2/(2\sigma^2))$ where $\sigma = 1.0$ and $\boldsymbol{x}_i$ is the position of data point i. Figure 9 shows that the input data is successfully clustered into the 3 expected clusters.

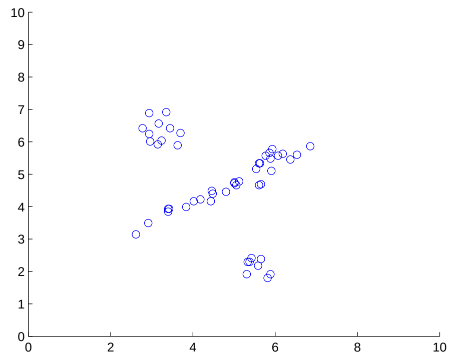


Figure 8: Input data for spectral clustering.

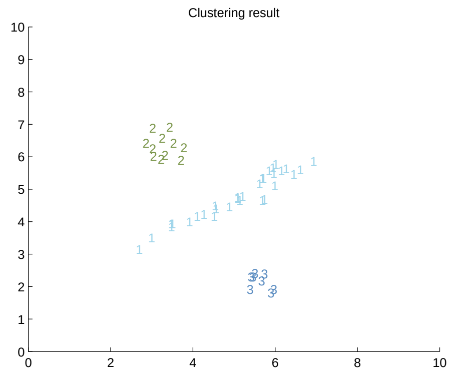


Figure 9: Result of spectral clustering.

4 Conclusion

We have presented a description of our proposed visualization methods, and described some of the computational machinery needed to implement it. The preliminary results have confirmed that these tools are valid for the visualization of small synthetic graphs. A physically-based graph layout algorithm was described which can be used to draw graphs on embedding surfaces has been developed. In the future this method can be used to draw graph nodes and edges directly, or be used to position the local deformations and material properties which will represent graph properties. Code for performing hyperbolic mapping has been developed which will be used in the user interface implementation. A clustering method has been implemented and tested. Such a clustering method is important to our multiscale approach to visualization. Future work towards development and testing the proposed methods on real STEP datasets will involve the following tasks:

- STEP parsing and large graph generation.
- Multiscale graph decomposition using clustering.
- Embedding surface formulation.
- Preliminary user interface development.

References

[1] Andrew Witkin: Physically Based Modeling: Principles and Practice, Particle System Dynamics, SIGGRAPH '97 course notes.

[2] Andrew Witkin: Physically Based Modeling: Principles and Practice, Constrained Dynamics, SIGGRAPH '97 course notes.

[3] W.E. Lorensen and H.E. Cline: Marching Cubes: a high resolution 3D surface reconstruction algorithm, Computer Graphics, Vol. 21, No. 4, pp. 163-169 (Proc. of SIGGRAPH), 1987.

[4] Paul Bourke: Polygonising A Scalar Field, http://local.wasp.uwa.edu.au/~pbourke/geometry/polygonise/

[5] John Lamping and Ramana Rao: The Hyperbolic Browser: A Focus+Context Technique for Visualizing Large Hierarchies, Journal of Visual Languages and Computing, vol. 7, no. 1, pp. 33-55, 1995.

[6] Tamara Munzner: H3: Laying Out Large Directed Graphs in 3D Hyperbolic Space, Proceedings of the 1997 IEEE Symposium on Information Visualization, pp. 2-10, 1997.

[7] Mark Phillips and Charlie Gunn: Visualizing hyperbolic space: unusual uses of 4x4 matrices, SI3D '92: Proceedings of the 1992 Symposium on Interactive 3D graphics, pp. 209-214, 1992.

[8] Ulrike von Luxburg: A Tutorial on Spectral Clustering, Statistics and Computing, vol. 17, issue 4 (December 2007), pp. 395-416, 2007.

[9] Marina Meila and Jianbo Shi: Learning Segmentation with Random Walk, Neural Information Processing Systems, NIPS, 2001.

[10] Jianbo Shi and Jitendra Malik: Normalized Cuts and Image Segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence(PAMI), Vol. 22, No. 8, pp. 888-905, 2000.