A Survey of Some Recent Results in Comptuer Graphics

Robert Patro

May 19, 2006

Introduction

The field of computer graphics is moving forward at an exponential rate. Even as GPU processing power increases at rates exceeding those predicted by Moore's law, the size of the datasets being acquired and rendered is increasing even faster. As a result, much research has been done recently which deals with making these very large datasets manageable. Work has focused on methods to help filter, view, render, and generally process vast amounts of visual data. For example, high precision laser scanners have brought about the rapid acquisition of very dense point cloud data sets. However, these very acquisition methods often result in models that suffer from a degree of unwanted noise. A "simple and fast" method to filter out such noise, yet preserve high frequency features, was presented by Fleishman, *et al.*[1]. Furthermore, with such an abundance of large models, it is necessary to develop automated methods by which they may be optimally lit and viewed. To this end, Lee *et al.*[3], have created a system for automatic light placement and silhouette edge and proximity shadow generation, which allows for the greatest exposition of surface features and details. In a similar vein, Lee *et al.*[5] also developed a new saliency metric that attempts to measure low-level perceptual significance. This concept of mesh saliency promises to have wide applications in the field of graphics. Finally, as a result of the realization of the increasing growth of the $\frac{compute}{bandwidth}$ ratio of GPUs (thus implying an equivalent increase in the growth of the $\frac{modelsize}{bandwidth}$ over the $\frac{modelsize}{compute}$ ratios), a novel approach in the vein of the stream processing paradigm is presented by Kim *et al.* [6]. This method allows for the factoring of large models into vertex and transformation (effectively translation at this point) streams which can then be recombined at the Vertex Processing Unit (VPU), effectively trading VPU computation for bandwidth. Each of these methods in their own right, represent significant progress forward in the effort to bridge the gap between the rapidly increasing size of the datasets acquired and our ability to visualize them.

Bilateral Mesh Denoising

Current methods of automatic data acquisition have lead to very dense datasets. However, along with the ability to discern ever smaller details within a scanned object, the problems posed by noise have grown more prominent. Because the devices used are more sensitive than ever, they are more subject to high frequency noise that causes inaccurate discernment of the actual underlying surface. Traditional filtering methods are effective at removing high frequency noise, however, in the process they often remove small scale details in the data. Filtering high-detail, dense models with traditional filters in many cases nullifies the benefits of the higher resolution acquisition techniques. To tackle this problem, Fieishman *et al.*[2003] suggest the use of a bilateral filter, a method which has proven highly effective for feature preserving denoising in image processing. The bilateral filter acts as a Gaussian filter, but introduces an extra term into the filter's kernel to account for differences in the intensity domain between local samples. They extend the bilateral filter into 3-space and introduce methods to avoid the two main complications **shrinkage**, and **drifting**.

In two dimensions, the bilateral mesh filter works on a local neighborhood of samples. A normal Gaussian filter, acting as a convolution kernel, weights points only in accordance with their distance from the center of the kernel, such that we can define the closeness in the distance domain as $W_c(x) = e^{\frac{-x^2}{2\sigma_c{}^2}}$ where x is the distance between two given

samples. The bilateral filter introduces a second term to account for the difference of samples in the *intensity* domain, we can view this intensity difference term as another Gaussian $W_s(x) = e^{\frac{-x^2}{2\sigma_s{}^2}}$, where x here is the difference in intensity between two samples. The terms σ_c and σ_s are parameters of these equations which may be set according to user preferences, or chosen dynamically based on the input data. The introduction of this intensity term to the kernel allows for the removal of noise while attempting to preserve edges and details. Fleishman *et al.* attempted to create a 3D analog of the 2D bilateral filter. While the concept of distance is easily extended from 2D to 3D, an attempt must be made to discover an analog for concept of intensity. A 2D bilateral filter, as used in image processing, considers the difference between two samples in the intensity domain as the difference in the gray-scale values between those two samples. Fleishman *et al.* propose considering intensity in 3D as the signed distance of a sample from the tangent plane of the mesh at the vertex which is at the kernel's center, such that if v is the vertex for which we are currently processing the bilateral filter and P is the plane tangent to the mesh at v [1], then for $v_1 \in N(v) \neq v$ they define the intensity difference $I(v) - I(v_1)$ as the distance from P to v_1.

Intuitively, one can imagine that the distance domain term $W_c(x)$ has the effect of smoothing the surface while the intensity domain term $W_s(x)$ has the effect of maintaining high-frequency data which has intensity similar to that of the sample being processed. The overall effect of applying this bilateral filter to the mesh is the removal of noise, due to the smoothing term, and the maintenance of high-frequency detail, due to the intensity term. Furthermore, the filter can be applied in an iterative fashion, with each iteration attempting to remove more noise while maintaining features. Since the application of the bilateral filter may lead to shrinking of the model, they use the exact volume preservation technique presented by Desbrun *et al.* [1999] to scale the model after filtering. Furthermore, the averaging effect in the distance domain can lead to a problem known as vertex drift which may move a vertex away from its original location, and as a result reduce the regularity of the mesh. The method presented here avoids vertex drift by only moving the vertex v which is being processed along the direction of it's normal, thereby eliminating vertex drift.

While the bilateral mesh filter produces convincing results, it has some shortcomings. The assumption is made that the data sets and meshes to be processed are regular, that is for a mesh M, $|N(v_1)| \approx |N(v_2)|$, $\forall v_1, v_2 \in M$. If neighborhoods contain too few samples, then the bilateral filter may have ill effects, possibly eliminating features which have been mistaken for noise. Fleishman *et al.* acknowledge this issue and point out that this problem does not exist in image processing, since images are regularly sampled. Yet, they propose no solution, and make the assumption of approximate regularity for the meshes processed by their algorithm.

Geometry Dependent Lighting

With the ever increasing complexity of geometric models, it is becoming easier for small features and details to get marginalized during rendering. One of the main ways to expressively render such details is through lighting. Light placement allows the viewer to comprehend certain parts of an image or rendering. Lee *et al.*[3] provide a system

[1] The normal of this plane is given as the weighted average (by triangular area) of the triangles in the 1-ring of v

called *Light Collages* by which lights can be automatically placed around a model so as to expose features and details of a model such as "local surface orientation, curvature, silhouettes, and fine texture"[4]. Further, they take advantage of globally discrepant[2] lighting to make features even more noticeable than might otherwise be possible.

The *Light Collages* system determines where lights should be placed around a given model, with the goal of exposing the most visual detail in the dataset. The system uses curvature as a metric to determine the manner in which geometry should be lit. Thus, the first step of applying such a procedural lighting model is to segment the mesh into patches based on local curvature. Once the model has been properly segmented, the light directions are chosen by maximizing a light placement function, $P(\vec{l})$. This function consists of two separate weighting functions $S(i, \vec{l})$, the specular weighting function, and $D(i, \vec{l})$, the diffuse weighting function. Both the specular and diffuse lighting components play an important role in determining how model details are accentuated. Specular highlights have a tendancy to intensify distinct shape in regions of high curvature, a desirable characteristic. Yet, such highlights could easily overwhelm subtle details present in areas of low curvature, and hence lighting placement that results in specular highlights on low curvature areas should be avoided. Likewise, the diffuse lighting function assigns values to a given vertex v_i, so that the value of $D(i, \vec{l})$ at v_i is similar to the curvature intensity at v_i[3]. The value of the general light placement function $P(\vec{l})$ in direction $\vec{l}$ is given as $\sum_i (S(i, \vec{l}) + D(i, \vec{l}))$. Higher values of the light placement function denote "better" lighting directions, and so the best n lighting directions from a large set of possible directions can be chosen according to this function[4]. After the "best" lighting directions are chosen, these lights are assigned to the patches of the model via an iterative threshold scheme. The lights are assigned to patches in accordance with a similarity function $E(p, l_k)$[5]. The value of the similarity function for the current "best" light l_1 and a patch p_k is computed, if the function yields a value below a certain threshold, the patch is assigned to the given light, and the contributions to the light placement function of l_1 by the patch p_k is then deducted. Once light placement has occurred for every patch in the model, the illumination between neighboring patches is blended to eliminate visual continuities.

Beyond the visual cues created via specular and diffuse lighting, the *Light Collage* system provides a mechanism for creating proximity shadows and silhouette edges to enhance depth perception around certain regions of a model. These proximity shadows and silhouette edges allow for the illustration of distance (different depths) between close portions of a model lit with similar intensity. Such similarities in intensities yield close pixel colors in the rasterized image which lead to difficulty in discerning boundaries between the different sections, even when significant spatial distance may exist within the data. The proximity shadows and silhouette edges produced by the *Light Collages* system attempts to eliminate this problem, thus augmenting depths cues which are not considered in the light placement function that might otherwise be lost to the viewer

[2]It has been shown that globally discrepant lighting is not easily noticed by the viewer, and in many cases globally discrepant lighting can provide compelling feature exposition.

[3]Here the curvature intensity at v_i is given as $c_i = \frac{(\kappa_i - \kappa_{min})}{(\kappa_{max} - \kappa_{min})}$, where κ_i is the mean curvature at vertex i, κ_{min} is the minimum curvature value across the entire mesh, and κ_{max} is the maximum curvature value across the entire mesh

[4]Lee *et al.* compute $P(\vec{l})$ for 12,000 uniformly distributed lighting directions

[5]Given a patch p, its associated set of surface points S_p, a light l_k, and $I_i(l_k)$, the illumination intensity at vertex $v_i \in S_p$ attributed by l_k, the similarity function is defined as $E(p, l_k) = \sum_{i \in S_p} (I_i(l_k) - c_i)^2$

after the process of rasterization.

The *Light Collages* system represents a very large step forward in the area of geometry dependent lighting and procedural light placement. The system produces visually compelling results which bring forth details and local surface cues of model geometry that are simply lacking under more traditional lighting schemes. Lee *et al.* even present a spherical harmonics based method to approximate the light placement function[4]. This allows for significantly faster computation of the light placement function and for much lower storage requirements if the light placement map is to be pre-computed. The *Light Collages* system represents a fairly comprehensive work in the area of procedurally generated light placement, but it leaves room for the inclusion of other factors into the light placement equation. Some visual cues, such as color and reflectivity, which are highly connected with lighting, are not currently considered in the light placement function, and their inclusion may lead to even better results.

Mesh Saliency

Even as the size and complexity of models increases, the human viewer still has an incredible ability to simplify the model in their mind by extracting and weighing "important" features. The identification of some of these features is based on the context in which the viewer encounters the subject of the model in real life, or in some other way on the higher level semantics of the model. However, in many cases, the attribution of significance to certian parts of a model is based on lower level visual cues. In developing mesh saliency, Lee *et al.* attempt to calculate a quantitative measure of the low-level visual properties that makes something perceptually significant to a viewer. Previous methods in this vein of research used a simple geometric quantity such as curvature, which, while important, is itself a global measure incapaple of accounting for the local context of a detail or feature. The mesh saliency metric is based on a measure of curvature, but also incorporates a center-surround mechanism to help acquire some sense of local context. By acquiring and incorporating such local context, the saliency metric provides a measure that not only appears more correct visually, but also has the desired characteristic of being scale dependent[6].

Lee *et al.* provide a method that calculates saliency at each vertex of a mesh according to the curvature[7] of that vertex in conjunction with the properties of a local neighborhood of points[8]. First, by computing the Guassian-weighted average of the mean curvature over the neighborhood of points within a certain distance of v, they are able to discern the local context in which v exists. Next, by computing this Guassian-weighted average at multiple (fine and coarse) scales[9] and calculating the difference in the computed values, they are able to see how the curvature at v varies with respect to the mean Guassian-weighted curvature of its neighbors. Thus by carring out this calculation, Lee *et al.* aim to first define the local context of a vertex in terms of the weighted mean curvature of its neighborhood, and then to interpret the mean curvature of that vertex at a local,

[6]By scale dependent, it is meant that what is considered significant at one scale may not be deemed so at another scale.

[7]Here, the term curvature refers to $\mathcal{C}(v)$, the mean curvature of v. Where κ_1, κ_2 are the principle curvatures at v, then $\mathcal{C}(v) = \frac{\kappa_1+\kappa_2}{2}$

[8]The neighborhood of a vertex v, denoted $N(v, \sigma)$, is given by $N(v, \sigma) = \{x | \|x - v\| < \sigma, x \text{ is a mesh point } \}$[5]

[9]The effect of computing the average at multiple scales is achieved by varying the size of the neighborhood (σ) of v considered in the Gaussian-weighted average.

context-aware, scale, rather than at the global scale normally assumed by curvature calculations. This new, local, and context-aware curvature metric for the vertex is now defined as the vertex's saliency. This saliency measure is much more closely coupled with perceptual significance than the original curvature metric; this can be understood through a simple example. One could imagine a high detail model of a freshly cut lawn on which a newspaper is lying. In this situation, the blades of grass exhibit sharp edges and areas of extremely high curvature, while the newspaper, by comparison, represents an area of relatively low curvature. However, in this scene, it is the newspaper and not the individual blades of grass that draw the viewer's attention. While a simple curvature metric would place emphasis on the high curvature blades of grass, the saliency metric will place empasis on the newspaper due to its difference in mean curvature from its surrounding neighborhood. Similar circumstances occur often in models, and in such cases this saliency measure is efficient in determining the perceptual significance of certain features against a background noise of curvature.

The concept of mesh saliency is very basic, and as a result, very powerful. Even in the original paper[5], Lee *et al.* used saliency to enhance mesh simplification and automatic viewpoint selection. When aiding in mesh simplification, computing a saliency map for a mesh allowed certains parts of the mesh to be marked as perceptually significant and consequently they were be simplified less than their surrounding areas. Saliency guided simplification produced more visually pleasing results than did the chosen competition[2]. Furthermore, when applied to the problem of automatic viewpoint selection, maximizing the visible saliency often produced more "natural" and informative viewpoints for models than did the previous approach of maximizing visible curvature (though this was not always the case). Saliency promises to be an important tool in future research because it provides a new and superior way to attribute importance to regions of a model, and such a notiion of importance can allow more selective and intelligent operations and computations on a mesh.

Vertex Transformation Streams

As the processing power of GPUs has increased at an exponential rate, the bandwidth allowing the transfer of data to the GPU has quickly fallen behind. Since bandwidth has now become the major bottleneck in the rendering of large datasets, a desirable action would be to allow for computation at the VPU (or even PPU) to replace the transfer of extra data, thus trading computation for bandwidth. Kim *et al.*[6] present a novel method for factoring a model into two separate streams representing sets of data points and the transformations that relate them. On optimal data, it can be shown that a model consisting of n vertices, traditionally requiring $O(n)$ data to be sent to the GPU, can be factored into $O(\sqrt{n})$ vertices and $O(\sqrt{n})$ transformations, thus requiring only $O(\sqrt{n})$ data be send to the GPU. These streams are then recombined at the VPU to reproduce the original data. The goal is to factor out the most common translations relating the vertices in a model.

Kim *et al.*, present the concept of a vertex pool[10], a set of vertices which *cover*[11]

[10]The "pool" terminology is common, just as a thread pool represents a set of threads that may be reused to accomplish different tasks, thus removing the overhead of new thread creation, a vertex pool represents a set of vertices that, under a given translation, cover another set of vertices which need not be sent separately to the GPU.

[11]A vertex v_1 is said to cover another vertex v_2 under transformation t if $t(v_1) = v_2$

other vertices under a given translation. Thus the process of factoring the model into its component streams is reduced to the problem of finding the largest pools and their corresponding translations. They make the observation that pools can be made larger by allowing certain vertices to be excluded from the mappings of given transformations. Thus, overall, it may be beneficial to include a given vertex v_i in a transformation pool, but mark it inactive under a transformation t_j, if this allows more vertices to be included in the pool an active under the given transformation. Kim *et al.* have experimentally verified that on average, vertex pools are optimal when about 50% of the vertices in a pool are made to be active under a particular translation. Because of the limits of current generation graphics hardware, the greatest speed benefits result from only a few (2-3) of the largest pools and their corresponding transformations being sent to the GPU. The results show that the largest two pools cover about 80-85% of the vertices in the model. Any vertices not covered by a pool and translation are marked as *singleton* vertices and must be sent independently to the GPU.

Once the model has been effectively factored into vertex pools and corresponding transformation streams, the pools and streams can be sent to the GPU. Here, a vertex shader can apply the proper translations to the vertices from the source pool to produce the set of vericies in the model covered by the source vertices under the given translation. Thus the original data is effective reconstructed on the GPU from the input streams. The savings in bandwidth are between 200% and 300% and as a result of the marginalization of this bottleneck, framerates were observed to increase by about 30%.

The research done on vertex transformation streams appears to be a monumental step in a very promising direction. The concept of factoring a model has numerous benifits, perhaps the most obvious of which is the bandwidth which can be saved by sending the factored streams of the data to the GPU, rather than the model as a whole. Future research may eventually allow for the general processing of such factorizations, thus possibly reducing expensive operations done on large data sets by a factor of $O(n)$. While the methods and concepts presented by Kim *et al.* are novel and highly promising, they are highly restricted specializations of more general themes. They take into consideration only translations on vertices, and ignore the more general set of transformations. Furthermore, the factorization into vertex and transformation streams fails to incorporate alot of other information that is normally associated with data points such as normals and colors. If such methods are to be widely used, these directons should be considered for future research to discover more general solutions in this promising direction.

Conclusion

The rapid acquisition and generation of large data sets has lead research in a direction that allows us to more easily handle the ever increasing size and number of models. For example, while current methods of high resolution data acquisition are sensitive to noise in the measurement and the ambient environment of the object being scanned, methods have been developed[1] to allow for the removal of such noise while simultaneously preserving high frequency features of the data. Furthermore, the magnitude of current and future datasets threatens to make certain features and details within these datasets difficult to visualize. Lee *et al.*, however, have created an automated lighting system which optimally places lights (in a possibly globally discrepant fashion), silhouette edges, and proximity shadows, so as to allow the viewer to discern surface features and properties of the model. Lee *et al.* have also provided a concept that defines salient regions of a mesh, and a new

way in which to compute this saliency metric. This work looks promising, and will in many cases allow for automatic detection of perceptually significant (at least with regard to low-level visual cues) regions of a mesh. Finally, Kim *et al.* have explored the factoring of large models into vertex and transformation streams in an attempt to marginalize the GPU bandwidth bottleneck by trading computation for bandwidth. Further research into this concept appears very promising as it may drastically reduce the amount of data which must be processed when performing certain operations on a mesh. As the size and number of models available increases at an exponential rate, computer graphics research must provide ways to process and visualize these data sets so that they are manageable to the user and become a benifit and not a burden.

Bibliography

[1] Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. Bilateral mesh denoising. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003)*, 22(3):950–953, 2003.

[2] P. S. HECKBERT and M. GARLAND. Optimal triangulation and quadric-based surface simplification. *Computational Geometry*, 14:49–65, 1999.

[3] C. H. Lee, X. Hao, and A. Varshney. Light collages: Lighting design for effective visualization. pages 281 – 288, 2004.

[4] C. H. Lee, X. Hao, and A. Varshney. Geometry-dependent lighting. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 12, No. 2:197–207, 2006.

[5] C. H. Lee, A. Varshney, and D. Jacobs. Mesh saliency. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005)*, 24, No. 3:659 – 666, 2005.

[6] Youngmin Kim C. H. Lee and A. Varshney. Vertex transformation streams. *Graphical Models*, (to appear), 2006.