Approximating Scatterplots of Large Datasets Using Distribution Splats

Matthew Camuto[1], Roger Crawfis[1], and Barry Becker[2]

[1]The Ohio State University - Department of Computer and Information Science - Columbus, OH 43210
[2]SGI - Mountain View California 94043

ABSTRACT

Many situations exist where the plotting of large data sets with categorical attributes is desired in a 3D coordinate system. For example, a marketing company may conduct a survey involving one million subjects and then plot peoples favorite car type against their weight, height and annual income. Scatter point plotting, in which each point is individually plotted at its corresponding cartesian location using a defined primitive, is usually used to render a plot of this type. If the dependent variable is continuous, we can discretize the three-dimensional space into bins or voxels and retain the average value of all records falling within each voxel. Previous work employed volume rendering techniques, in particular, splatting, to represent this aggregated data, by mapping each average value to a representative color.

However, if the color mapped attribute is categorical, then this technique is inadequate, since we can not represent or calculate an average value to which a single color is assigned.

This paper presents a new technique called distribution splatting. Our method is not only faster than traditional scatter plotting for massive data sets, but also visually preserves the sample density within the volume. The main problem examined in this paper, is the representation of all the attributes at a voxel using a single splat. We represent a splat with a multi-colored tessellated hexagon. The number of elements, opacity and size within each splat can be modulated with respect to record size to show the sample density accurately. Other techniques such as percentage thresholding and an evidence viewing mode are presented as extensions. Examples from real data sets are presented to display the effectiveness of our method.

Keywords: Information Visualization, Categorical, Nominal, Splatting, Volume Rendering, relational data

1. INTRODUCTION

As warehousing of relational data becomes more pervasive, it becomes increasingly difficult to plot the accumulation of information using standard reporting techniques. Traditionally, scatter plots have been used which place a primitive for each data point in a 3D coordinate system. However, this method has disadvantages. Any technique, which tries to represent each point with a unique graphical primitive, will exhibit problems with overplotting and rendering efficiency, as the number of data points grow. A large data set, rich with valuable information, may not yield anything valuable if undesirable occlusion problems are present or poor interactive frame rates occur. By using points alone, the user does not get a feel for the record size and the overall sampling density of possible regions in question.

Becker [1997][1] presented a method for the volume rendering of relational data in liu of using conventional scatter plotting techniques. This solution bins and aggregates the data and stores it in a new relational table. Each column in the table represents a unique variable while each row corresponds to a unique bin. Binning a particular variable helps reduce the overall record size. For example, in a survey we may want to create bins for *age* in ten year increments starting at age 0. Using this method, all records where $age = [10, 20)$ will be binned together. The number of records (also termed *weight*) is also associated with each row in the table after binning is performed. If no binning is performed the weight of a particular row will be one. By binning, we can reduce a data warehouse containing millions of records to a sparse n1 x n2 x n3 grid of values for interactive viewing.

Further author information:
Matthew Camuto: camuto@cis.ohio-state.edu
Roger Crawfis: crawfis@cis.ohio-state.edu
Barry Becker: becker@engr.sgi.com

For Becker's implementation, any continuous or categorical variable, can be mapped to the x,y and z axes and be treated as independent variables. Numerical attributes may be mapped to external animation slider dimensions. Continuous variables mapped to axes or sliders must be binned first. There is at most one aggregate for each combination of axes and slider values. The color assigned to an aggregate is determined by computing the average of the numerical value mapped to color and looking up the corresponding colormap value. If categorical variables are mapped to axes, they can be ordered alphabetically, by weight, or by some other attribute. Ordering the values by their correlation with the attribute mapped to color appears to be best.

The data is then rendered using the volume rendering technique known as splatting, as presented by Westover.[2] Volume rendering can provide faster speeds then traditional scatterplotting for large data sets. In order to emphasize areas with a larger record weight, the opacity of each splat is set as a function of the record weight. The weight is often just a count of the records, but it may be any arbitrary weighting of records, hence weight is used rather than count.

While Becker discusses the problem of volume rendering of relational data having scalar data values, it does not address the problem of mapping a categorical data value to color as the dependent variable. In this research, we address the problem of how to plot large numbers of records in a scatterplot, where each value of a categorical attribute has been mapped to a color. For example, given a survey of individuals favorite fruits, we may want to examine the eating preferences for people with respect to their residential state, income and weight. If we have a set of 520 samples of people between the ages of 30 to 35, living in Arkansas and making between 45,000 and 50000 annually and we observe that out of these 520 samples, 260 people prefer strawberries, 200 prefer bananas, and 60 mangos. In traditional scatter plotting each point could be individually plotted at its cartesian coordinate location (Arkansas would be mapped to an enumerated value). However, in volume rendering the data would be binned into a single voxel that has a center mapping to one particular 3D coordinate location. In this case a single bin would exist representing the sample population consisting of people between the ages of 30 to 35, living in Arkansas and making between 45,000 and 50000 annually.

This paper addresses the issues associated with rendering the distribution of values stored at each voxel. First the data is aggregated into a set of bins (or voxels) and then the graphical primitive is then drawn at each bin location to represent the distribution of data in that aggregate. Wong [1996][3] has a similar approach, but is still limited to showing numerical attributes. The approach here involves storing a distribution at each voxel of the aggregation. At each voxel the proportion of each category present is stored along with the overall record weight.

Overall, little research has been conducted on volume rendering of vector-valued attributes. Crawfis and Max [Crawfis93][4] presenting an effective technique for representing three dimensional vector fields. This work extended splatting, by adding anisotropic detail within a texture computed to represent the footprint function. We will also use a special splat primitive to show the distribution of nominals within a voxel. The technique we describe has been incorporated into a currently unreleased version of MineSet, a decision support product from SGI[Brunk97].[5]

2. METHOD

Our overall goal is to generate a useful visual representation of a volume, in which each voxel represents a distribution of categorical attributes. With this scheme we want to render the volume and accurately show the contribution of each of the n categorical values. This includes selecting a specific and unique color for each of the values. The amount of visual representation a particular categorical value receives also needs consideration. The representation could be based upon total percentage contribution to the volume of a particular category, or some other criteria, depending on the desired effect. Currently we use n maximally separated (in hue space) random colors. There may be better ways to assign colors. For instance, a clustering technique could be used to determine proximity of categorical values, then similar values could receive similar colors. This might be particularly useful if the number of values is large.

We use a splat based volume rendering method, in which each voxel is rendered with a tessellated splat. This splat is multi-colored, with the percentage of each color being proportional to the percentage of each nominal value present within the voxel. We also present ways to: threshold out categorical values with a small percentage contribution, remove regular and distracting patterns, emphasize regions where local distributions vary from the global distribution, and accurately depict differing voxel record weights.

2.1. Splat Representations

When rendering the volume, we need a splat representation that not only can give an accurate visual cue to the categorical distribution, but will also not introduce artifacts and/or distracting patterns. Artifacts such as incorrect blending or repetitive geometric patterns, if introduced, may prevent the user from extracting important trends from the rendered scene. The main goal is to define a splat primitive that can accurately represent the distribution of the vector valued data within a particular voxel, while avoiding any distracting or regular patterns. Since each splat represents a vector valued entity, we subdivide it into a set of basic primitives. We have developed a tessellated hexagon shaped splat for this purpose. A hexagon provides a good base, since it can be easily subdivided into equilateral triangles. Given a desired level of subdivision n, a hexagon can be divided in $6 * \sum_{i=0}^{n}(1+2i)$ equilateral triangles. With this scenario, every sub-region (triangle) within the splat will has same surface area. This will not over emphasize any particular sub-region within our splat. We have developed two possible representations, one being semi-transparent with the other being completely opaque.

2.1.1. Semi-Transparent hexagonal splat representation

The hexagon template is comprised of equilateral triangles and is shown in Figure 1. This particular hexagon has 384 triangles within it. The percentage of each value within a particular aggregate determines the percentage of triangles used to represent that aggregate when rendered. Using the fruit preference example as a reference, we would use 50 percent of the triangles to represent the 260 people who prefer strawberry at that particular voxel. The color representing strawberry is the same for every voxel, despite the ranking compared to other fruits.

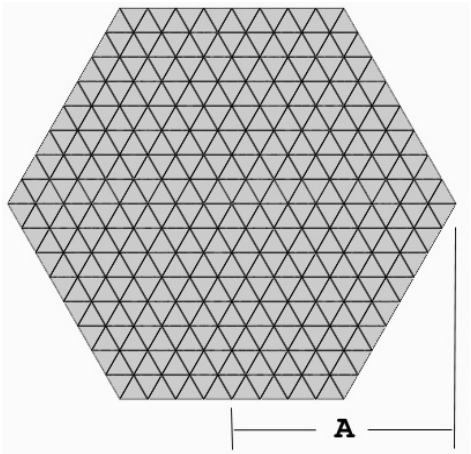
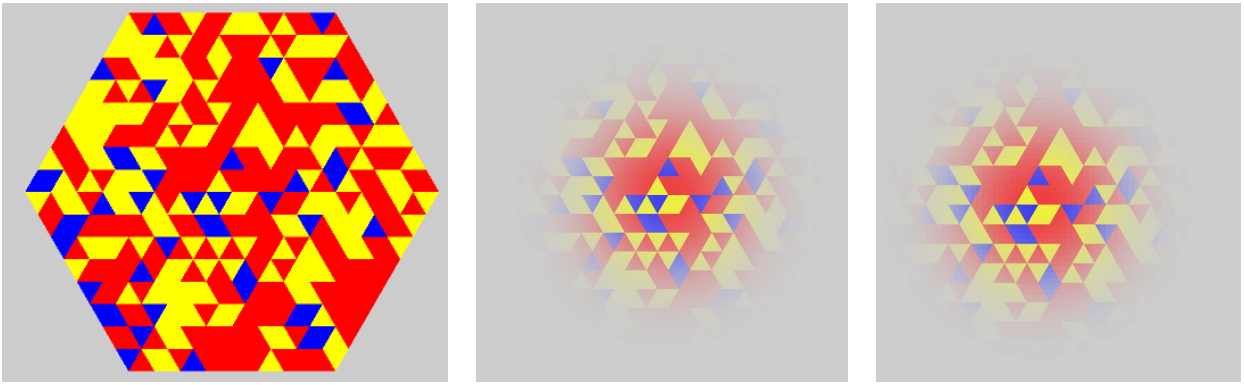


Figure 1. Hexagonal Splat Template.

For the semi-transparent splats we set the vertex opacities of each vertex using the optimal reconstruction kernel presented in [Crawfis93].[4] This kernel modulates the opacity based on radial distances between 0 and 1.6 from the splat center. To obtain a similar opacity modulation to that of a uniform colored splat, the distance 'A' in Figure 1 has a dimension set to 1.6 units. When rendered, the opacity of each triangle is linearly interpolated between each vertex. Traditional color texture mapping, where the opacity over the surface is modulated with an opacity map, can also be used for a smoother interpolation. Since we use many Gouraud shaded triangles within a splat, there is no advantage to texturing these small triangles. The change in opacity along a triangle edge is very small and no artifacts from linear interpolation are detectable (See Figures 2b and 2c).

Figure 2 shows a multi-colored splat with and without opacity modulation. The color distribution is proportional to that of the fruit example above with strawberry, banana and mango mapping to red, yellow and blue respectively. Figure 2(a) shows a multi-colored splat before vertex opacity modulation. Figures 2(b) and 2(c) show the splat after opacity modulation. Figure 2(b) image shows the splat where the opacity is linearily interpolated based upon vertex opacity values. The vertex opacity values were calculated based upon the optimal kernal. Figure 2(c) shows the same triangle after opacity modulation using an opacity texture map based on the optimal kernel with $128x128$ pixels. Note how Figures 2(b) and 2(c) images appear the same. Figure 2 also appears on the color plate at the end of this paper.



(a) Full Hexagon Template

(b) Opacity Modulation Using Linear Vertex Interpolation

(c) Opacity Modulation Using Texture Map

Figure 2. Basic Semi-Transparent Splat Primitive [See Color Plate]

2.1.2. Opaque hexagonal splat representation

Semi-transparent splat kernels produce renderings that appear 'fuzzy' or blurred to the user and may also produce excessive incorrect color blending. This blending occurs when incorrect colors appear in the rendered volume after the compositing operation. For example, a transparent yellow over a transparent red will appear orange. This could possibly lead the user to wrong conclusions about the data. The problem could be further aggravated in nominal splatting where we have a large number of colors, not only within the volume, but inside each voxel.

To avoid this situation, we present a splat, composed of many small non-overlapping opaque triangles. The triangles within this splat primitive have centroids in the same locations as the semi-transparent splats. However, the size of any particular triangle is reduced based upon the distance between the splat center and the triangle's centroid. The scaling factor s, of each triangle is calculated with the function:

$$s = 1 - \frac{r^2}{r_{max}^2 + \epsilon} \tag{1}$$

where r_{max} is the maximum distance from the splat center among all the triangles within the splat, ϵ is some small number relative to r_{max}, and r is the distance from centroid of the current triangle being rendered to the splat center. Note that the range of this scaling factor is between 0 and 1. The triangles will always be disjoint and the small spaces between them will allow for viewing deeper into the volume. Figure 3 shows a splat of this type. This scale factor should be further multiplied by the splat density (see below).

2.2. Splat Density Modulation

Each bin within the volume may have a different weight of data that falls within it. However, we may desire a voxel containing 1000 samples to be more apparent to the user then one containing 12 samples. This can be achieved by further modulating the splat densities based on these record counts.

As explained in [Becker97],[1] we can modulate the overall splat opacities or triangle sizes by ρ according to the density function:

$$\rho = 1 - e^{-c*w} : 0 \leq \rho \leq 1 \tag{2}$$

Here c is a user defined constant that can be adjusted with a slider at run time. The record weight is represented by w. By including w, we can place a visual emphasis on voxels that have more samples. A user may be more confident with the data in a voxel with more samples, thus it should be more pronounced within the volume.

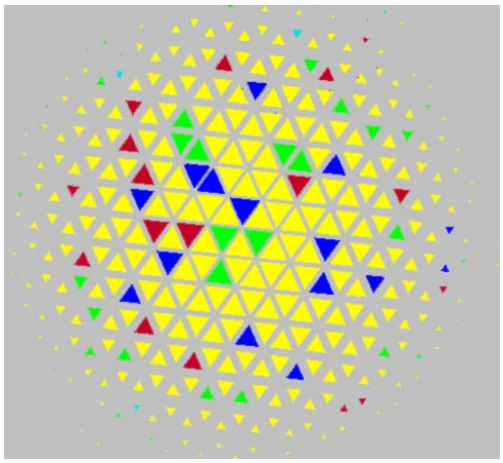


Figure 3. Basic Opaque Splat Primitive [See Color Plate].

When using the semi-transparent hexagonal splats the opacity at every vertex is further modulated by ρ. In the case of the opaque splats, the size of each triangle is further scaled by ρ. This still keeps the triangles completely opaque while putting less emphasis on voxels with low sample counts.

2.3. Regular Pattern Removal

Regular patterns produce features which can attract the eye as much or more then the relevant trends in the data. To avoid these patterns we first randomize the ordering of the triangles to be rendered. This will randomly distribute the color (as much as possible) over the voxel so we get a more uniform distribution of colors. Furthermore, we randomly rotate the textured splat about the eye direction to remove any recurring patterns.

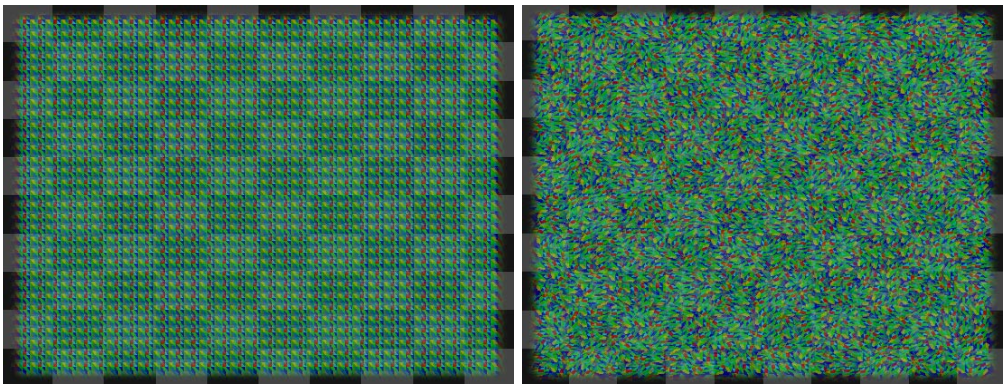


Figure 4. 2D Slice of a volume

Figure 4 shows a slice of a 2D volume with and without regular pattern removal. All the splats in the left image have the same orientation while the splat on the right have been randomly rotated about the eye direction. Note that the right image does not display these regular patterns.

2.4. Constant Splat Rendering

Sometimes within the volume many voxels exist that have only one categorical value to render. This could occur in situations where only a few categories exist or where there is a high correlation between axis values and the categorical values being mapped. If only one categorical value occurs within a single splat, the rendering method presented here is not efficient. To avoid rendering excessive triangles, a simple single color Gaussian splat is rendered

when only one categorical value is present. This will cut down on the rendering time per splat, and will thus speed up the entire scene rendering time.

2.5. NULL Thresholding

Datasets, regardless of size, can have a plethora of categorical values to choose from. Thus many colors may occur in the rendered scene. We wish to present the strongest trends and reduce the amount of noise in the representation. We achieve this by thresholding out categorical values that do not make a strong contribution to the final rendered image.

We provide a proportion threshold, below which nominal values will be colored a neutral color (such as gray). This is particularly useful if there are many values, making it more costly to render. Also, it is difficult to perceive the distribution if too many disparate colors are present in a single splat. In order to reduce visual clutter or extraneous information we may want to interactively filter out data based on user defined thresholds. Thresholds can be based on items such as overall record weight, percentage of a particular nominal in a voxel, etc. For example, we may want to remove voxels with a small record weight because they may not give a good sample representation. These records may be insignificant to the user and also may slow rendering time. We may also limit the display to the top n nominals per voxel. This is especially useful when a large pool of potential nominal values exist. Figure 5 shows the same splat from Figure 2 with a 15 percent threshold value. Note that the color Blue, representing mangos (which had an 11.5 % contribution) is no longer present in the splat, and a NULL color (gray) replaces its triangles.

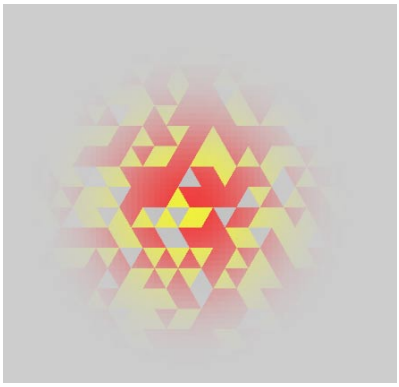


Figure 5. Basic splat primitive with a 15 % NULL threshold [See Color Plate]

2.6. Evidence Viewing Mode

If the overall distribution of categorical values is very skewed in the dataset, it may be very hard to detect regions where rare values occur more often than expected. For example, suppose you map "race" to color using the census dataset. The majority of people are white, so the color assigned to this value dominates the visualization. What you really want to see is places where local distribution varies significantly from the global distribution. This is difficult to detect if most of the data is a single value (or a small subset of values).

To emphasize local variation, it is possible to divide the local distribution by the global distribution and normalize the result. This new distribution then contains normalized conditional probabilities for each value. That is the relative probability of a datapoint being in that bin, given that it has a particular value for race. This technique is the same as used by Naive Bayes classification models [Good97,Domingos96,Matroni99].[6][7][8]

For example, suppose the distributions at a given bin and the overall sample population of ethnicities in the dataset that appear in Table 1.

In the basic approach described, this splat will be colored 70 % white, and 20 % will be allocated to the other races. Overall, the scene will be 84 % colored the color corresponding to white. If you are interested in finding regions having a greater inclination of Hispanics, it will be hard to differentiate between 2 very small percentages. If we normalize the splat distribution as described above, the new distribution of splat colors will be as in Table 2.

Ethnicity	Bin Distribution	Global distribution
White	70 %	84 %
Black	5 %	8 %
Asian	25 %	5 %
Hispanic	0 %	3 %

Table 1. Sample data distribution for ethnic population.

Ethnicity	Bin/Global Distribution	Contribution to Splat
White	70/84 = 0.833	12.9 %
Black	5/8 = 0.625	9.7 %
Asian	25/5 = 5.0	77.4 %
Hispanic	0/3 = 0.0	0.0 %

Table 2. Sample normalized data distribution for ethnic population.

As you can see, using this approach the splat will be colored 77.4 % Asian because the local Asian population is 5 times greater than predicted by the overall distribution. We are thus not visualizing the local distribution, but the amount of deviation from the global distribution.

2.7. Slider Animation

Even though we are rendering vector valued data which is multi-dimensional, we still have only three independent variables if we use a cartesian coordinate system. However, we may want to examine the categorical data with respect to other independent data as well. For example, suppose we are examining automobile preferences (categorical value) and have bins with age, occupation and income mapped to the X,Y and Z coordinate axes. We may also want to see how the data changes with respect to other independent variables, such as peoples weight or height. Unfortunately, we can not have more than three coordinate axes.

In order to achieve animation across external dimensions, we needed to develop a method to interpolate between distribution splats using a slider. First we bin the slider variable into distinct intervals. Each bin position defines a unique splat plot. The interpolation method described in the example below is very similar to that described by Becker [1997],[1] but here we are interpolating between distributions instead of scalars. The following is an example of the "tricky" case where we don't store the full vector for each bin, because the percentage weight threshold has caused most values to be lumped under an "other" value (to be colored gray in the visualization). Note that in the interpolated result there are more unique values stored than at each endpoint.

Lets say your animation slider variable is "year manufactured" and that for a given splat position you have the distributions at 2 adjacent slider positions displayed in Table 3. The distributions the percent concentration within a bin and the weight is the number of records within that bin. Now suppose we want to find the distribution for the splat that will appear when the slider is moved half-way between these two discrete bins. This will involve weighting the two bins according to the parametric interpolation value, summing these values and then normalizing the data with respect to record weight in order to get the new percent distributions for the categorical data. Table 4 shows this progression for the two positions mentioned in Table 3. The final row of this table, displays the approximate percent distribution for each type of car at the interpolated value. These values were obtained by dividing the appropriate values in row 3 (*Sum Rows*) by the averaged record weight of 150. This is the data that will ultimately be used for distributing the colors in the splat upon rendering. When the slider is moved between marked bin positions, the scene will show an interpolation based on the real data at adjacent locations.

This interpolation technique extends easily to two or more external animation sliders. For 2D or 3D slider extensions, we use bilinear or trilinear interpolation in conjunction with the nominal splat interpolation.

3. RESULTS

The first example we present uses a simple dataset about cars. In this example, the data has been binned and aggregated to produce a small number of voxels. We map miles per gallon (mpg_bin), horsepower (horsepower_bin)

Position	Buick	Nissan	Honda	Pontiac	VW	bmw	other	weight
1	20	30	0	0	10	0	40	100
2	10	0	10	20	0	40	20	200

Table 3. Bin distribution for two adjacent slider positions.

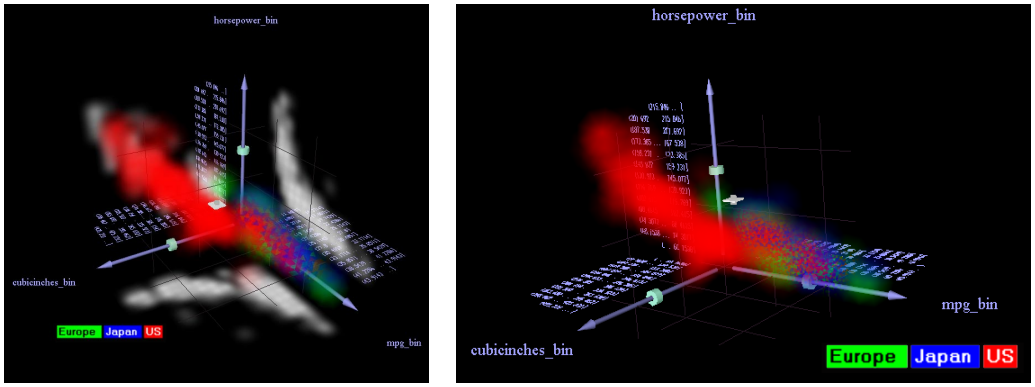
Step	Position	Buick	Nissan	Honda	Pontiac	VW	bmw	other	weight
Weight Rows	1	1000	1500	0	0	500	0	2000	50
	2	1000	0	1000	2000	0	4000	2000	100
Sum Rows	NA	2000	1500	1000	2000	500	4000	4000	150
Normalize	NA	13.3%	10%	6.6%	13.3%	3.3%	26.7%	26.7%	150

Table 4. Steps to interpolate bin distributions for two adjacent slider positions.

and engine displacement (cubicinches_bin) to the 3 coordinate axes. The categorical attribute we would like to map to color is the *origin* of the car. In this example, we have three origins of car: *European*(green), *Japanese*(red) and the *United States*(blue).

Figure 6 shows two rendered views of this data set using the semi-transparent splat primitive. Figure 6(a) has contains 96 triangles with Figure 6(a) having 384. These images were generated using the semi-transparent hexagonal based splats with a sub-division of 384 triangles. Figure 7 is identical to Figure 6 but uses the opaque splat primitive.

From these figures, using this particular data set, we see that the United States produces the cars with the largest engine sizes and horsepower ratings and these two attributes are somewhat linearily related. Here it is apparent that all three car origins have models that have low horsepower and smaller engines. Many of the splats in this region have a distribution that involves all three car origins.



(a) 96 Triangles

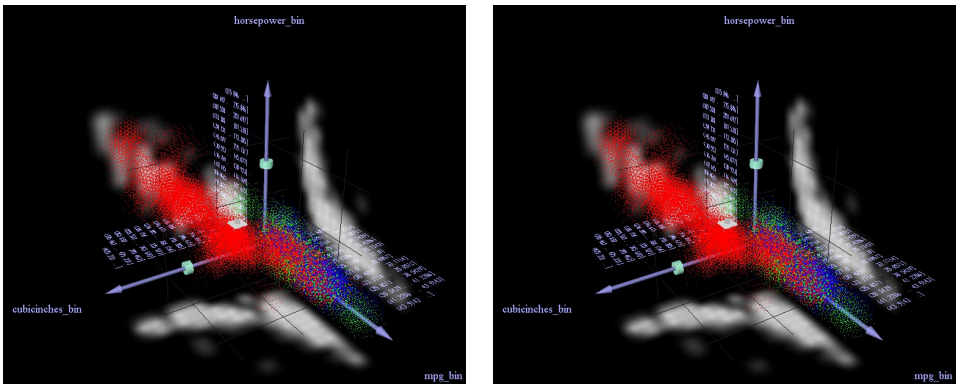
(b) 384 Triangles

Figure 6. Cars Dataset With Semi-Transparent Splats

The next example is taken from a U.S. *census* dataset from 1994. In this particular study the 3D independent coordinate axes map to age (age), capital gains (capital_gain) and gross income (gross_income). The categorical variable being rendered in this example is *relationship*. There are six different categorical values for relationship: *Husband*(blue), *Not-in-family*(teal), *Other-relative*(purple), *Own-child*(yellow), *Unmarried*(green) and *Wife*(red).

Upon initial examination it is clear that a high population of those in the survey have low capital gains. This is clear by the dense data population in the lower region of the graph and is visible in the left image of Figure 8.

However, by adjusting the weighting factor for opacity, more information about the dataset can be obtained. The images of Figure 9 show images rendered with small and large opacity weighting factors. Note that the Figure 9(b)



(a) 96 Triangles (b) 384 Triangles

Figure 7. Cars Dataset With Opaque Splats

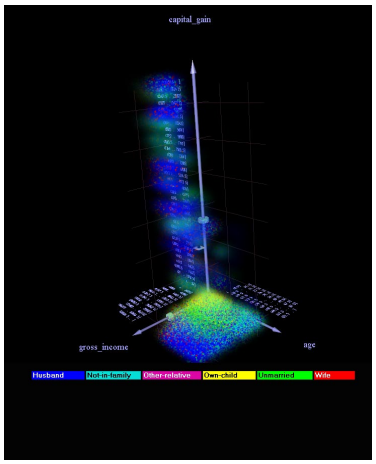


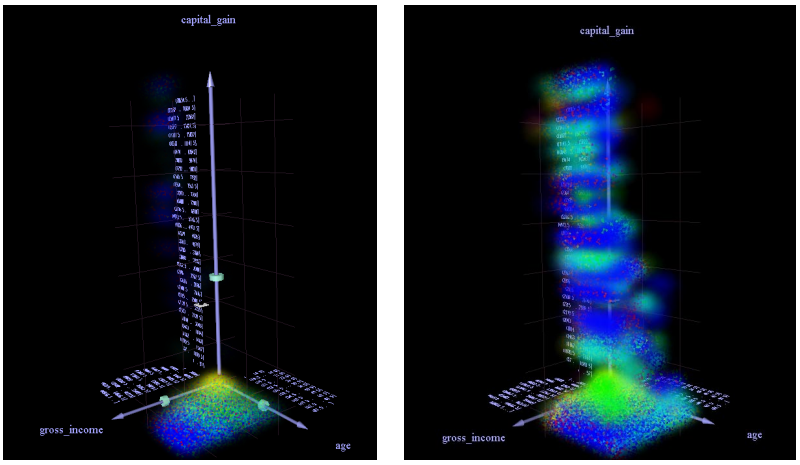
Figure 8. Census Dataset

displays more data within the volume then originally visible. These new visible voxels have relatively low weights within this volume.

By examining the pictures in Figure 8 it is also apparent that the data is comprised mainly of husbands and unmarried subjects with a lesser amount of people whom are not in the family being present. From these plots it is apparent that, from those surveyed, unmarried people generally had smaller capital gains while most of the people with high capital gain and a large gross income were husbands.

Note that wives make up a small percentage of the sample pool. This does not give a good representation of a the potential for a wife to have a large gross income or large capital gain. Evidence viewing mode should be used here to display regions where wives have a chance to occur.

Figure 10 represents this volume as viewed in evidence mode. Note now that the wives are much more visible. This shows regions where more wives are more likely to appear had more people in this category are surveyed. From this mode it is clear that both husbands and wives have about the same chance of falling within certain regions. Also note that people who are unmarried have a smaller chance to report large incomes compared to husbands and wives. This could possibly indicate that many men and women reported joint income, which would probably be higher then those who were unmarried, thus creating the higher correlation between the two, while over shadowing the income of the unmarried.



(a) Small Threshold Constant (b) Large Threshold Constant

Figure 9. Census Dataset With Differing Thresholds

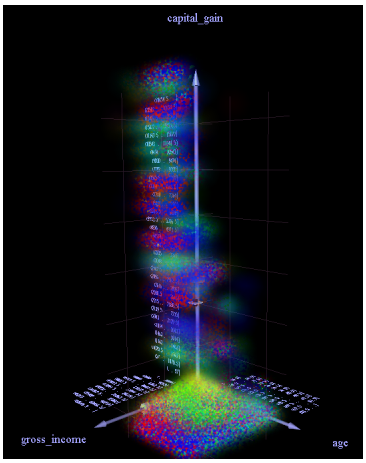


Figure 10. Census Dataset in Evidence Mode.

4. CONCLUSION AND FUTURE WORK

This paper presented a novel procedure for the volume rendering of categorical data. This paper was a direct extension of the work presented by Becker.[1] We have presented a new splat primitive to handle the rendering of the nominal data. We have also presented methods for evidence viewing of such volumes and also the animation of these volumes with additional variables.

In the future, the effectiveness of various new splat primitives should be tested. The use of different types of textured splats or point cloud splats may yield promising new results.

Another extension would be to look at the rendering of multiple dependent variables (either scaler or nomimal) within the same volume. An example of this would be to render the temperature, energy and cloud density distributions along with wind velocity in the same volume. Another possibility is simultaneously viewing the concentrations of n different chemical contaminants bassing through a volume of earth.

As the number of records in a dataset increases, it is inevitable that some sort of aggregation will need to be performed in order to achieve a useful visualization. The result of this work is to demonstrate a method for producing a volumetric histogram of categorical data.

5. ACKNOWLEDGEMENTS

We would like to thank the Mineset group for help in supporting this project. We would also like to thank Naeem Shareef for proofing this paper many times. This work was supported in part by an NSF CAREER award #ACI_9876022.

REFERENCES

1. B. Becker, "Volume rendering of relationa data," in *IEEE Visualization*, IEEE Computer Society Press, 1997.
2. L. Westover, "Footprint evaluation for volume rendering," in *Proceedings of SIGGRAPH*, vol. 24, pp. 367–376, 1990.
3. P. C. Wong, A. Crabb, and R. Bergeron, "Dual multiresolution hyperslice for multivariate data visualization," in *Information Visualizations*, pp. 74–75, 1996.
4. R. Crawfis and N. Max, "Texture splats for 3d scalar and vector filed visualization," in *IEEE Visualization*, IEEE Computer Society Press, 1993.
5. C. Brunk, J. Kelly, and R. Kohavi, "Mineset: An integrated system for data access, visual data mining, and analytical data mining," in *Proceedings of KDD*, 1997.
6. I. J. Good, *The Estimation of Probabilities: An Essay on Modern Bayesian Methods.*, MIT Press, 1965.
7. S. Matroni and H. Vanderberg, *MineSet 3.0 Reference Guide.* SGI.
8. P. Domingos and M. Pazzani, "Beyond independence: Conditions for optimality of the simple bayesian classifier.," in *Machine Learning, Proceedings of the 13th International Conf*, pp. 105–112, 1996.