VISUALIZATION LAB WEB SUPPORT

PROJECT DOCUMENTATION

By

Chris Driggers

Supervised by

Dr. K. R. Subramanian, Department of Computer Science

May 5, 2001

SECTION 1.0

PREFACE

1.1 ACKNOWLEDGEMENTS

I would like to thank my wonderful wife Amy for her endless love and support during my tenure as a student. I would also like to thank K.R., Dr. Bashor, and Dr. El-Kwae for all of their assistance throughout these projects.

1.2 INTRODUCTION

This purpose of this documentation is to offer support to the work I have done during my senior year in the hope of providing a basis for future work on these projects. The paper is broken into three major sections. Each section is devoted to a specific project that I was involved in. Each section is broken into subsections containing background information, implementation details, and future work that should be done. The first project was a website for the Graphics, Visualization, and Imaging Group at UNCC. The second project was a motor neuron circuit simulator for Dr. Subramanian and Dr. David Bashor of the Biology Department. The final project was preliminary work on an image feature extraction and analysis program with Dr. Essam El-Kwae and Dr. Subramanian.

SECTION 2.0

GRAPHICS, VISUALIZATION, AND IMAGING GROUP WEBSITE

Chris Driggers
May 5, 2001
University of North Carolina at Charlotte

2.1 BACKGROUND

The Graphics, Visualization, and Imaging Laboratory in the Department of Computer Science was established for the development of new courses as well as enhancement of existing courses in the areas of computer graphics, visualization, and imaging. The lab provides the necessary infrastructure for research and development of new applications, conversion and digitization of audio and video from different applications, and processing and visualization of scientific/engineering data sets. Dr. Subramanian wanted a common website for colleagues to be able to access information about the various projects that are under examination by the group. Previously, each faculty member would use their own website to distribute information about their projects. This was inconvenient for people who are looking for information about the interesting research that is conducted at UNCC.

2.2 IMPLEMENTATION

The solution I proposed provided a website that had an associated database that housed the content for the website. The reason for this architecture was to eliminate the need for creating separate pages for each project. It also makes the addition of projects much easier, because page links do not have to be changed manually. It is also convenient to provide browsing functions to browse through projects for a specific faculty member or student. The ColdFusion markup language (CFML) was perfect for this application. It allows for the creation of web pages that can access databases to provide

dynamic content to viewers. The College of Information Technology now has a ColdFusion enabled server that was used to host the web site.

The database was constructed in the following manner:

Faculty table:

Id	name	link
5	My Name	http://www.…

A table row contains a unique numeric identifier, the faculty member's name, and an optional link to the faculty member's website.

Student table:

Id	name

A table row contains a unique numeric identifier and the student's name

Projects table:

Id	name	sp1	sp2	st1	st2	st3	st4	des	links

A table row contains a unique numeric identifier, the name of the project, the id number of a faculty sponsor, the id of a second faculty sponsor, the ids of up to four students assigned to the project, a long description of the project, and finally any supplemental links for the project.

The web pages use the CFML language to perform queries on the database and format the data for output as standard html that is viewable through any web browser. The website uses frames throughout to provide a uniform appearance to the viewer. The index.html page defines the typical frameset. There is a header frame that contains the graphic logo, a sidebar that contains the

navigation buttons, a footer that contains links to various UNCC websites, and the main frame that is used to display content. Visitors may:

- Browse through the projects.
- Select a faculty member from the faculty page in figure 1 to see their specific projects.
- Select a student from the students page, which is similar to figure 1, to see their projects.
- View a description of the facilities that are available.

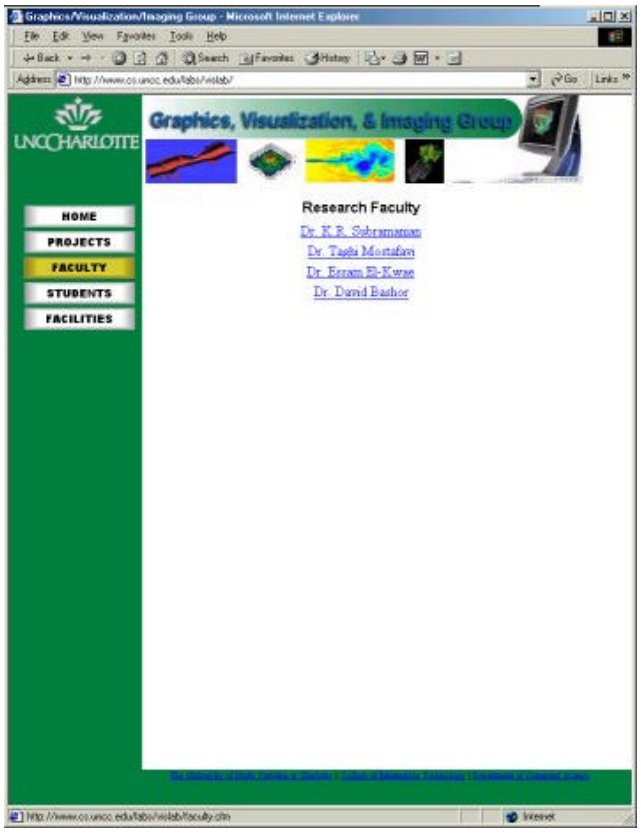


Figure 1: Faculty Page

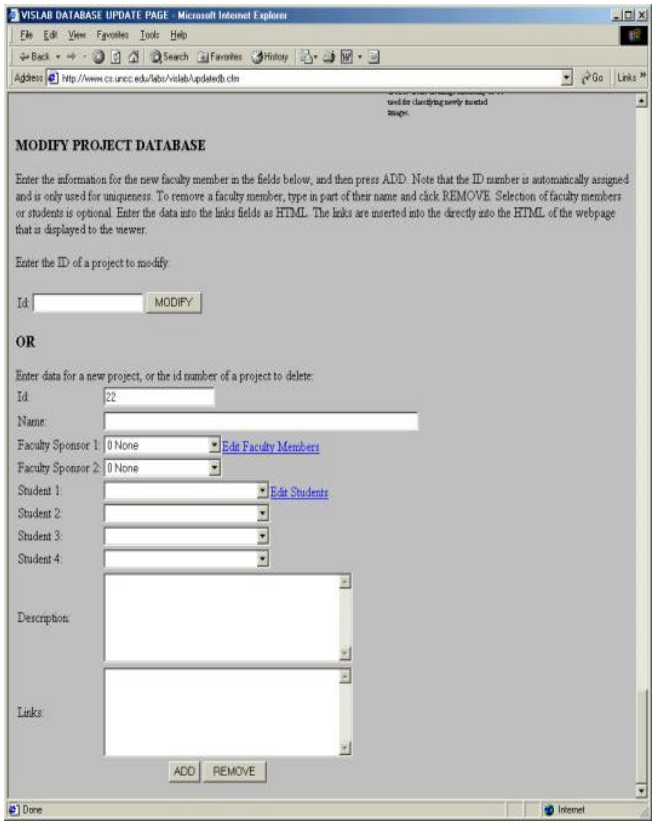


Figure 2: Update Page

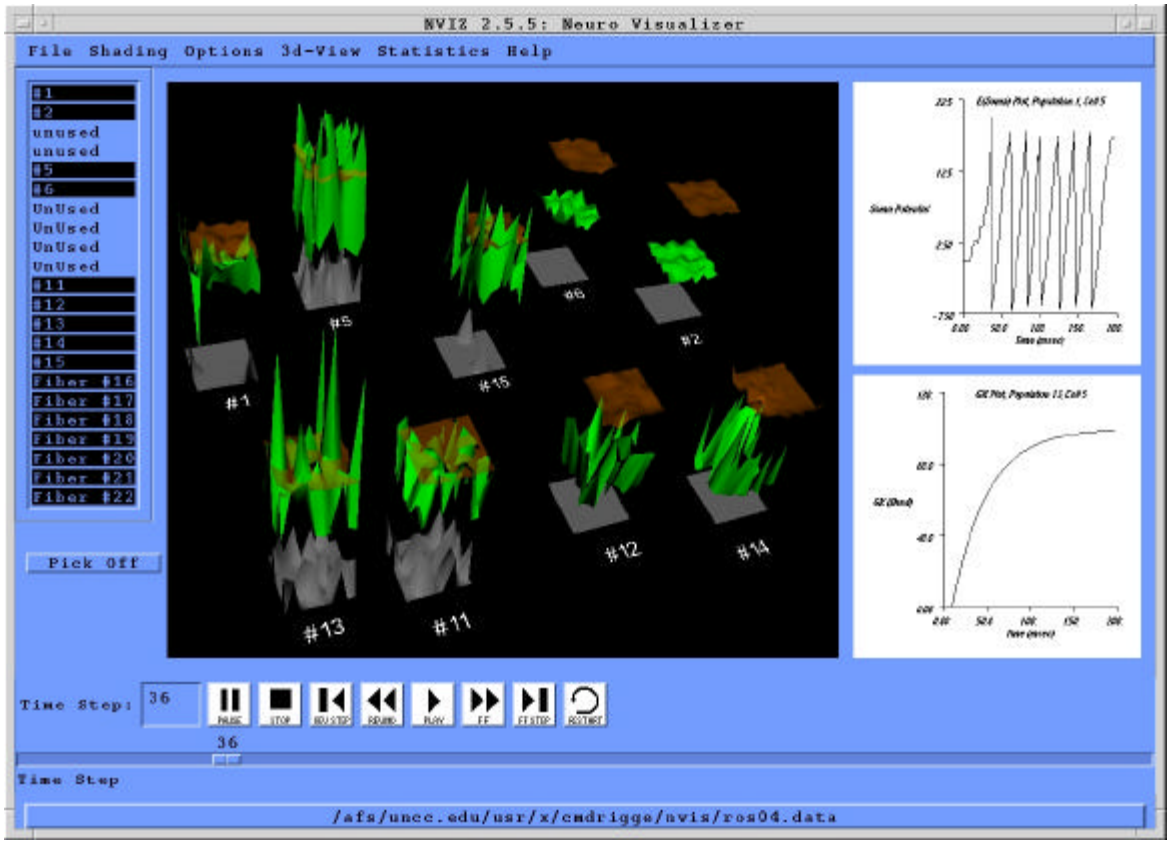
I have provided several pages that can be used to update all information in the database. Updatedb.cfm, shown in figure 2 above, is the main entry point to update projects. Access is also provided to the updatefaculty.cfm, updatestudents.cfm, and modifyproj.cfm pages. Projects, faculty members, and students can be added, removed or modified using these pages. This will make it easy to add future projects and maintain current and past projects. Please consult the source code of Apppendix A for further implementation details.

2.3 FUTURE WORK

This project was done during a transition period where I had to wait several months for the ColdFusion server to become operational. After which I discovered that the databases I had created with Microsoft Access were useless, because the school uses PostgreSQL as its database driver. I could not find a way to convert the data that I had made, so I had to create the database again in Postgres. This cost me some time, and I was unable to complete the website database to the extent that I would have liked. There are several projects underway that faculty members have no information available on the Internet for them. These projects will need to be added to the database in the future to provide a more complete representation of the work that is performed at UNCC. Also, a better facilities page needs to be provided that shows pictures of the labs and equipment that is available at UNCC.

SECTION 3.0

NVIZ 2.6 Neuro Visualizer User's Guide and Technical Documentation



Chris Driggers
May 5, 2001
University of North Carolina at Charlotte
Dr. David Bashor, Dr. K. R. Subramanian

3.1 BACKGROUND

NVIZ is a visualization tool that shows the circuit activity of spinal neural populations. It uses a mathematical model, based on the algorithms of MacGregor (1987), to generate a large-scale simulation of cell population interactions. Once run, this large amount of data is visualized in a 3D environment. The user then can view the populations at each time step and how they relate to each other. NVIZ greatly increased the rate at which data could be analyzed, and provides a graphical user interface for carrying out statistical analysis. For further background information, please consult James Harrison's paper in Appendix B. Please note that the name SPSIM has been changed to NVIZ.

The purpose of this section is to record all changes made to the previous version 2.0 of the NVIZ software. It provides instructions for using the new features of version 2.6, as well as the technical issues involved in implementing such changes. This will make the features easier for the end user to use, and for subsequent programmers to make revisions and additions. The statistical graphing functions of previous versions have been enhanced to use a built-in graphing mechanism, rather than the separate Xgraph program. The new features of version 2.6 include: built-in two-dimensional graphing functions, Save Image, Save Sequence, Make Movie, View Data File, and Print.

3.2 IMPLEMENTATION

The main goal of version 2.6 was to eliminate the need for using Xgraph to do two-dimensional statistical plots. The method to do this was to employ

vtkXYPlotActor to handle the generation of standard x-y plots. The original idea was to provide a new graphing window for each plot, so that a number of plots could be compared. This was handled by generating a new vtkXRenderWindow, and rendering the output of the XYPlotActor to the new window. The problem with this approach was that the newly created window would not refresh if covered by another window or minimized. Adding a vtkXRenderWindowInteractor to each instance of the plot window solved this problem, but created a new problem. Now, closing one of the child windows containing a plot would kill the entire application. Several emails to Kitware, the developers of VTK, and many hours of searching could not solve this problem. It was decided that the best course of action would be to rethink the graphing method.

A compromise was to create two small vtkXRenderWindows built into the main program interface beside the large three-dimensional view port. The process for creating a graph would allow the user to select which graph window, either upper or lower, to display the graph in. A vtkXRenderWindowInteractor was added to each of the windows so that when the user clicks on one of the small window with the left mouse button the selected graph is displayed in the large view port, and the simulation view would be moved to the small window that had been clicked. This allows for closer inspection of the data by the user, and allows use of the new print function. The process of the user creating any of the various statistical plots is the same, except that the dialog boxes now give the option of plotting to the upper or lower windows.

The method for generating the data for the input of the XYPlotActor involves creating a vtkFloatScalars, a vtkPoints, and a vtkPolyData object. The scalar and point objects are sent to the previously created functions that generate the statistical data. The scalar and point data are populated, and the data is written to a file as in previous versions. The outputting of the data to a file allows for further inspection with Xgraph, Excel, or other such programs. The scalar and point data is then used as input to the vtkPolyData object. The vtkPolyData object is then added as input to the vtkXYPlotActor object. Other formatting is then done to set the size of the plot in the window, font type and style, and turn the lines and points on and off. The graph is then rendered to the window selected by the user. The code for most of this operation can be found in the StatMenuCB.cc, Stat.cc, and neuroviz.cc files.

The swapping of the graphs from the small windows to the large one is a very simple process. The small windows were created in the neuroviz.cc code, and each has its own renderer and interactor. With this arrangement, all that needs to be done is to swap the vtkRenderer that is associated with a particular vtkXRenderWindow.

A late feature addition enables the user to view multiple datasets on the same graph. There is now a checkbox on each statistical graph dialog box that allows the user to select if they want to add the dataset to the graph that is currently in the window that they choose, or to create a new graph. If the user chooses to add to the existing graph, then the new dataset will be displayed in a different color. Currently, each graph may have up to six datasets associated

with it. This should be enough datasets for the user to see good results, more that six makes the data too crowded on the graph. A legend has been added to each graph that shows which population and cell that the dataset represents. The data in the legend is color coordinated with the data in the graph.

Since the VTK toolkit is still evolving, there are still a few problems with using the XYPlotActor. There is no efficient way of doing a plain bar graph for the histogram data using the PlotActor. This may require the creation of a specialized mechanism for creating the “bar” data representation. Presently, all that can be done is to approximate the data by tracing the outsides of where the bars would lie on the graph.

The File menu gives access to many of the new features of NVIZ 2.5.5. The Save Image, Save Sequence, Make Movie, View Data File, and Print functions are all accessed from this menu.

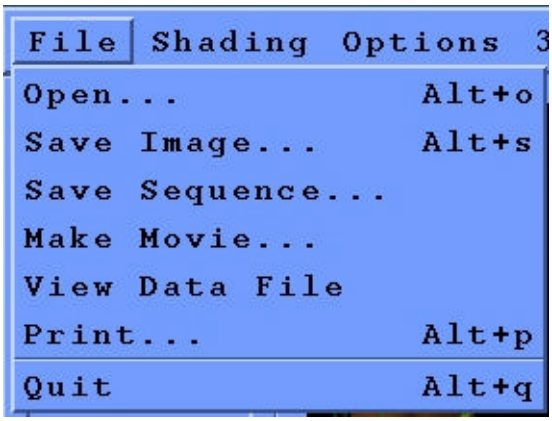


Figure 1: The File Menu.

The **Save Image** operation allows the saving of a single image. This will save a *.ppm image of the current main display window. This may be an image from the three-dimensional simulation, or one of the graphs that is being

displayed in the main display window. The default location is the current NVIZ directory, but any location can be specified. In the Save Image dialog box, select the directory, and type in the filename that you would like to save the image to. You do not have to type the .ppm extension, as it will be automatically added. Press the OK button to perform the saving procedure. The following figure gives an example of this procedure.

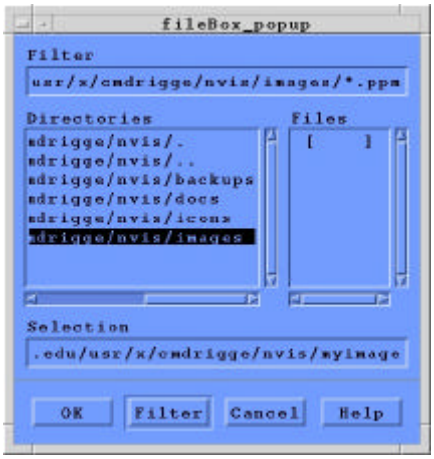


Figure 2: The Save Image Dialog Box.

The **Save Sequence** procedure allows the user to save a series of images from the current simulation. In the file template box, enter the path and filename template for the images. Next, specify the start, end, and step times. Press the OK button to save the sequence of images.

Figure 3: The Save Image Sequence Dialog Box.

The example above will generate 100 images in the nviz/images/ directory with the file names: myimage0.ppm, myimage1.ppm, myimage2.ppm, …., myimage99.ppm..

The **View Data File** menu item simply opens a text editor to allow the user to view and modify the input data. The only problem with this is that the program must be restarted to load a new data set. It is my understanding that this has been a problem for some time, and it lies somewhere in the cleanup of the simulation data. Once the data cleanup process is fixed, then the user will be able to edit the input data and start a new simulation without restarting the application.

The other main utility implemented in this version, is the ability to make QuickTime movies directly from the application. To make a movie, select **Make Movie** from the file menu. This will bring up the Make Movie dialog box. The dialog box is very similar to the Save Sequence box. The file template does not really need to be changed, unless another directory would like to be used for the temporary images. The last step is to specify the name of the resulting movie. It is recommended to save the temporary data, and the output movie in the /tmp directory of the local machine. This helps the process to go much faster, and lightens the load on the network. The user can then move the final movie to the necessary location. The process takes approximately 10 minutes per 100 frames to generate a 20 second video. The file size is about 5 megabytes per 100 frames at a 350 x 300 resolution.

Figure 4: The Movie Maker Dialog Box.

The movie creation process occurs in several steps. First, the images are saved to the temporary directory. To workaround the frame rate limitations of the dmconvert utility, 5 symlinks are created for each actual image that is saved. The images are then converted to jpegs, which is a compressed image format. This step reduces the final movie size significantly, with little loss of image quality. The series of images is then compiled into a Quicktime movie using the dmconvert media conversion program, with the animation (RLE) compression at 15 frames per second.

NOTE: It is very important that the main window of the NVIZ program is not covered by another window, or minimized during any of the image saving applications. This will produce artifacts in the output images, and the images will have to be saved again.

The **Print** utility prints the current image in the main display window. This may be an image from the simulation data, or one of the two-dimensional graphs. From a technical standpoint, the print utility in its current form is not a very elegant solution, but it is functional. To save printer ink/toner, some preprocessing must be done to the image. If the current display is the three-

dimensional simulation, then the background is changed to white, and all of the text is changed to black. The frame is then rendered with the inverted colors. A *.ppm image is saved in the /tmp directory with a temporary file name. The original text and background colors are then restored, and the image is redrawn. To the user, the process can be seen on the screen for a fraction of a second. A better solution to this would be to render to an off-screen render window, so that the user doesn't see the color change. The *.ppm image is then converted to a postscript file using imconv and sent to the printer by a lp command line operation. Again, this is a very simple solution that is functional. Further improvement on this method would be to create a postscript file directly from the NVIZ program. Currently, VTK only allows the saving of images as *.ppm files, but there may be other utilities out there that will write a postscript file from a vtkXRenderWindow.

3.3 FUTURE WORK

After many hours of work on this project, there have been many improvements. There is of course more work to be done to refine previous features, and add new functionality to enhance the usability and understanding of the data that is generated by the simulation. A number of items have been suggested. One such addition is to have a pure data analysis mode, where the user is only focused on the statistical data. The process used to create the two smaller graphing windows could be expanded to fulfill this data analysis mode. Another interesting proposition is to generate a three-dimensional model of the

spine, and show the neuron populations in their correct locations in the nervous system.

Bugs, or undocumented features, that need to be fixed:

- Creating a new simulation after one has already been performed. The data cleanup process needs to be fixed so that the user doesn't have to exit and restart the program to perform another simulation.
- The Rate Meter Plot for All Cells displays a new vtkXRenderWindow that kills the entire application if it is closed. This goes back to the problems that were documented earlier with using the vtkXRenderWindow as a child window. The solution I tried for this was to add the actor that is created in the graphing function to the renderer for one of the graph windows. The procedure worked nicely, but adding and removing the actor more than once would crash the application. I ran out of time to fix the problem properly, so I changed the function back to the old process. Users can minimize the window for the plot, but they just cannot close it until they are ready to close the application.
- Users can add multiple datasets to a graph. They can also add datasets of different types to a graph. This is not a problem because the data that is sent to the vtkXYPlotActor is of the same type. The problem is that the graph title, x-axis title, and y-axis title are completely overwritten instead of appended. For example, a soma plot can be added to a calcium plot. The title should be something like "Multi-Plot", the x-axis should be time, and the y-axis should be a combination of the y-axes for a soma and a calcium

plot. The legend should make the distinction of which dataset is of which type.

- If the program is to be expanded upon in the future, it will be necessary to get control of the memory and cpu requirements of this program. The entire program should be analyzed for memory leaks and ways to optimize the code to reduce overhead. One artifact of the mismanagement of memory is that after a large simulation has been performed, the movie making process will not work because there are not enough free resources to perform the data conversion.

One goal of this document was to provide the end user with instructions on using only the new functions of NVIZ 2.6. The user will have to consult other documents on how to use the basic functions of NVIZ, including: generating data files, running a simulation, and parameters for the various statistical graphs. The second goal of this document was to serve as a reference for the programmers who will follow in working on this project, and provide at least a foundation for understanding the evolutionary process of NVIZ from version 2.0 to 2.6.

SECTION 4.0

Content Based Image Retrieval from Image Databases

Chris Driggers
May 5, 2001
University of North Carolina at Charlotte
Dr. K. R. Subramanian, Dr. Essam El-Kwae

4.1 BACKGROUND

There has been an increasing interest in image databases in recent years. This is mainly due to the numerous applications in which images play an integral role. One of the unique features of image databases is their ability to retrieve images, similar to a query image based on the image content. The target of this project is to design and implement an image indexing and retrieval system based on a set of image features to be selected from color, texture, shape, and spatial constraints. This project performs some preliminary investigation to show the feasibility of using data visualization techniques to visualize the raw numerical data that is generated by performing extraction of several features from a series of images.

There have been many papers on this subject that provide techniques for performing image comparison based on low-level features. Indoor-outdoor images classification, city images versus landscape images, and photographs versus graphics have all been explored. The problem is that generally these papers only consider comparing one image type against another. The ultimate goal of this project is to be able to have a repository of image features that can be applied in any combination to achieve the best result when comparing any types of images.

In this paper, we apply the techniques explored in the paper by Athitsos, Swain, and Frankel to extract the features necessary for the classification of photographs versus graphics on the World Wide Web. A photograph is considered to be a scanned image or a digital photograph of a real-world scene.

A graphic is a computer drawn, man-made image that may represent a real-world object. There are a number of differences between graphics and photographs that can be exploited:

- Color transitions between pixels tend to follow different patterns. Photographs are of real-world objects that have texture, varying colors, and generally some noise. As a result neighboring pixels tend to not have the same RGB color values. Graphics on the other hand usually have large regions of constant color, since they are generated on the computer. Because of these large regions of color, edges in graphics are usually much sharper than they are in photographs. In photographs however, boundaries between objects are often blurred by poor focus and varying light, which produces a smoother transition.
- Highly saturated colors tend to appear more frequently in graphics than in photographs. Photographs depict real-world objects where highly saturated colors are rare.
- Graphics tend to have smaller sizes and more elongated shaped than photographs. Graphics are often longer in one dimension, and usually very small. Photographs are generally more square, and usually larger.
- Graphics have fewer colors than photographs because of the large regions of constant color they usually contain. On the Internet,

graphics tend to have very few colors in order to make them more compressible so that they will load faster into web pages.

We extracted the following image metrics as the paper suggests:

- The number of unique colors in the image. Graphics tend to have less than photographs.
- The prevalent color metric. We find the most frequently occurring color in the image. The value of this metric is the fraction of pixels of an image that have this color. Graphics tend to score higher because of their large regions of color, and their fewer numbers of colors.
- The farthest neighbor metric. This metric analyzes each neighbor of a pixel and defines the color distance d between two pixels as:

  $$d = |r-r'|+|g-g'|+|b-b'|$$

  Since color values range from 0 to 255, d ranges from 0 to 765. The farthest neighbor of a pixel is the pixel that produces greatest value of d. The final result is the percentage of pixels that have a distance greater than some threshold P. Since graphics tend to have sharp color transitions, for higher values of P graphics tend to score better.
- The saturation metric. Let m be the maximum and n be the minimum among the r, g, b values of a pixel. The saturation level is defined as p=|m-n|. The score of a image is the percentage of pixels that have a saturation level greater than or equal to some

threshold value P. Since graphics tend to have more saturated colors, for high values of P graphics will score better.

- The color histogram metric. A color histogram is a three dimensional array of 16 x 16 x 16 elements. Each pixel color (r, g, b) corresponds to a bin indexed by [floor(r/16), floor(g/16), floor(b/16)]. The histogram contains in each bin the fraction of pixels of the image whose colors correspond to that bin. We create a reference graphic histogram, and a reference photograph histogram by averaging hundreds of histograms of such images. The correlation between two histograms is:

$$C(A,B)=\Sigma_{I=0,\ I=15}\ \Sigma_{j=0,\ j=15}\ \Sigma_{k=0,\ k=15}\ (A_{ijk}\ B_{ijk})$$

  An image I has a histogram H_i . Let $a=C(H_{i,}\ H_{graphics})$ and $b= C(H_{i,}\ H_{photographs})$. The score s of the image is: s=b/(a+b). Photographs should score higher, since as $C(H_{i,}\ H_{photographs})$ increases, or $C(H_{i,}\ H_{graphics})$ decreases, s increases.
- Dimension ratio metric. Let w be the width, and h be the height of the image in pixels. Let m be the maximum and l be the minimum of w and h. The score of the image is (m/l). Graphics tend to score higher because of their more elongated shapes.
- Smallest dimension metric. The smallest dimension of the image in pixels.

4.2 IMPLEMENTATION

In Athitsos, Swain, and Frankel's paper, a copy of which is included in Appendix C, they construct a program using the metrics described above to make a decision about whether an image is a photograph or a graphic. We want to instead visualize the data that is generated by analyzing these features in order to determine graphically the best features for comparison. The metrics we have discussed each provide some single value score for each image. To begin the process, I wrote a java application to search through the Yahoo.com image gallery to download a set of photographs. I collected approximately 850 golf, football, and baseball images per the request of Dr. El-Kwae. The images are generally 24-bit jpeg image files of around 45,000 pixels. For graphics, I collected approximately 1000 images from a couple of sources. One source was a Gifart.com cd-rom that contains graphics specifically for use on the Internet. Another was entitled 25,000 web objects by Media Graphics International.

I developed a JAVA application to extract all of these features and output the scores for each metric to a data file that follows the Visualization ToolKit (VTK) unstructured grid dataset format. VTK is a set of programming libraries that provides tools for performing data visualization and analysis in a number of programming languages. The program performs two steps. First, it goes through a training step in which it constructs the reference color histograms for each type of image. The user specifies the length of the training phase. The second step is the actual feature extraction and analysis. Again, the user specifies how many

images to analyze. The program is written JAVA2 using JDK 1.3 and the Java Advanced Imaging API version 1.1 beta.

A second application written in the tcl/tk scripting language reads the data file and constructs a parallel coordinates plot of the data. A parallel coordinates plot allows for graphing multidimensional points on a single graph by using separate parallel axes. Examples of these types of plots are shown below in figures 1 and 2. These plots were generated using data from the feature extraction program. Moving from left to right, the axes on the graphs represent the following: number of colors, prevalent color metric, color saturation metric, smallest dimension metric, color histogram metric, farthest neighbor metric, and dimension ratio. Please consult the source code in Appendix C for further details of implementation.

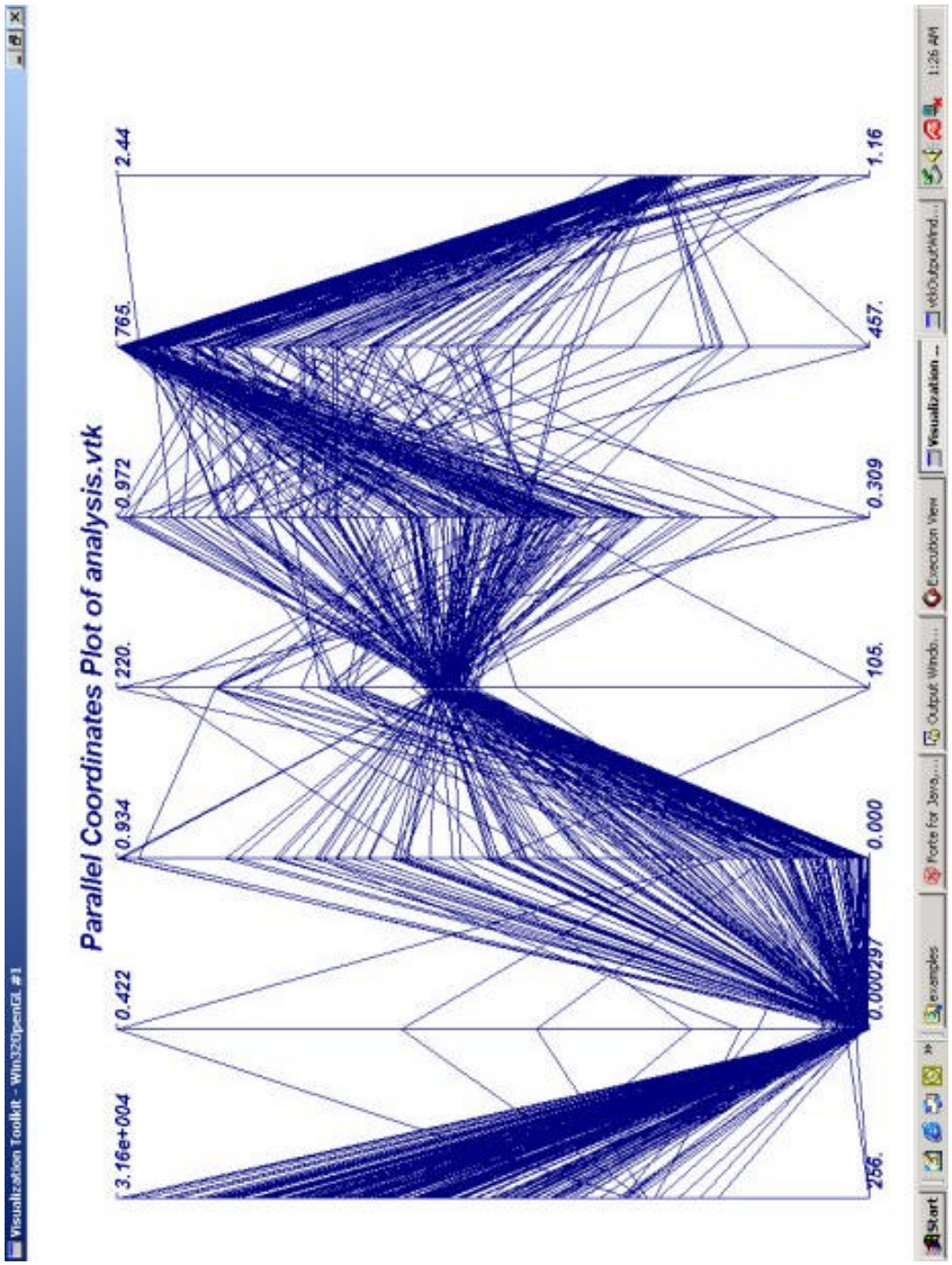


Figure 1: Analysis of a set of photographs
Axes represent (from left to right): number of colors, prevalent color metric, color saturation metric, smallest dimension metric, color histogram metric, farthest neighbor metric, and dimension ratio.

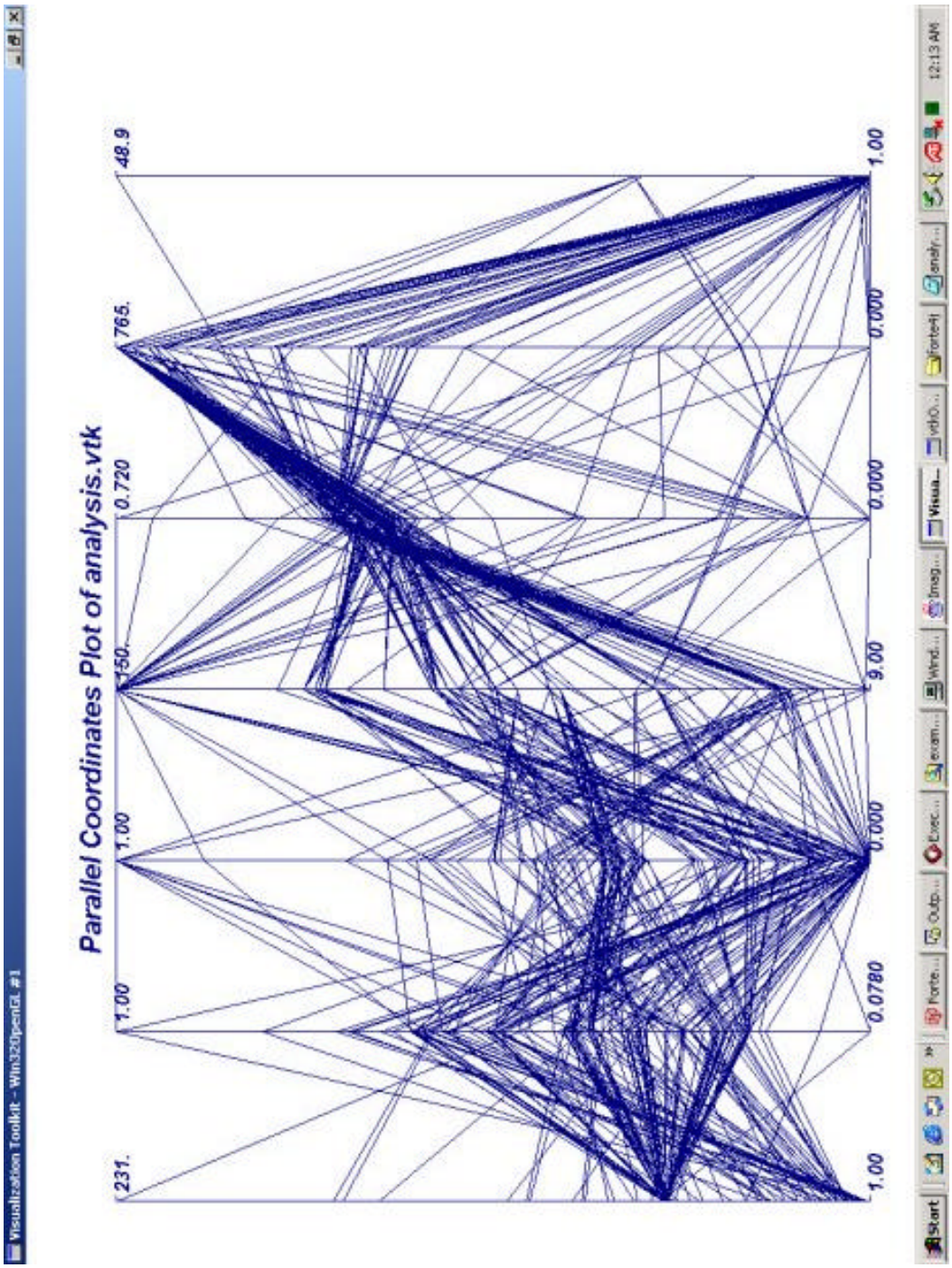


Figure 2: Analysis of a set of graphics
Axes represent (from left to right): number of colors, prevalent color metric, color saturation metric, smallest dimension metric, color histogram metric, farthest neighbor metric, and dimension ratio.

4.3 RESULTS

The graphs in figure 1 and figure two are from an analysis of 200 images with 200 training steps. One immediate result than can be seen by comparing the two graphs is that the number of colors metric on the first axis is excellent for providing a distinction between graphics and photographs. The graphics analyzed all had less than 231 colors, with most having between 1 and 64 colors. Also, the graphics tended to cluster in this metric, while photographs were fairly well dispersed across the axis. The photographs all had more than 256 unique colors, with a maximum of 31,000 unique colors. Thus, the graphics and photographs score ranges in this metric are completely separate, and provide a good measurement for determining if an image is a photograph or a graphic.

The next result that can be seen from these graphs is that the prevalent color metric on the second axis is also a good indicator of the class of an image. Graphics ranged from .078 to 1, with a majority being fairly evenly distributed around .30. Photographs had a much smaller range, as they ranged from .000297 to .42. Most photographs were clustered around the minimum value, while only a few went outside of this cluster up to the maximum. This is to be expected because of the high number of colors, and the various textures that are prevalent in a photograph. While the graphics usually had about 30% of the pixels with the same color due to the large regions of constant color that are apparent in graphic images.

The third axis is the color saturation metric with a threshold value of 63. The score of an image is the percentage of pixels that have a color saturation

value greater than or equal to the threshold. The graphs are inconclusive on this metric. Photographs were evenly dispersed across the axis, while graphics were more sporadic with middle and lower clusters. This is probably due to the fact that most of the photographs were richly colored sports images, and may have been post processed to give them a richer appearance for use in print media. The graphics were of various types, so I am unsure about their relatively low scores in this metric.

The fourth axis is the smallest dimension of the image. Here again we see that the ranges of values for the photographs and the graphics are disjoint. The photographs generally clustered around a smallest dimension of 200 pixels, while all of the graphics were fairly dispersed with a 150 pixels being the maximum of the smallest dimension. This is as expected, because photographs tend to be around the same size and are usually more square, while graphics are usually small, and of varying sizes. The disjoint sets show that this is a good metric for determining whether an image is a photograph or a graphic.

The fifth axis represents the color histogram metric. The photographs had evenly distributed scores on this metric. The graphic images were generally clustered around .50. The maximum value .972 in the photograph analysis shows a high correlation between the image under examination and the reference photograph histogram that was constructed during the training phase. The .50 average score by the graphics shows that the graphics were generally 50% correlated with both the graphics and the photographs histograms. We would expect photographs to score higher than graphics in this metric, and there are

many photographs that do score higher than the .50 average of the graphic images. Without having the same images to test my program against as Athitsos et al. did in their paper I am unsure if these results are correct, or if there is a problem with my algorithm. It could be that the images I used in my tests were too similar, and that a wider range of images needs to be collected in order to gauge an accurate reading of this metric.

The sixth axis is the farthest neighbor metric. Due to an oversight on my part, this axis represents the maximum color transition for an image. It should be the percentage of pixels that have a transition value above a certain threshold. The source code has been changed to reflect the correct measurement, however due to time constraints I was unable to run a new analysis for inclusion in this paper. Still, it can be seen that most graphics have a maximum transition value of 765, which is the maximum possible. On the photographs plot, you can see that many have a high valued color transition, but a greater portion of the images are spread across the axis. The difference that would be seen in a corrected graph is that while a photograph may have a pixel with a high transition value, it is not likely to have a high percentage of pixels with high transition values. Graphics on the other hand would have a high percentage of pixels with high transition values.

The last axis represents the dimension ratio metric. Photographs tended to have a dimension ratio of around 1.2, with a few having different values. The photographs were not square, but more rectangular as you would imagine a photograph to be. Unfortunately, the graphics plot had images that had very large

dimension ratios, which makes it hard to see the difference in the smaller values. The images with the large ratios were tiling backgrounds for a websites that were very wide, but had very small heights. Nonetheless, the values are spread evenly between 1 and 2, which shows that the graphics tended to be of varying sizes where one dimension was greater than the other, but not significantly greater.

4.4 FUTURE WORK

There is an enormous amount of work that needs to be done in the areas of image analysis and feature selection so that accurate systems can be constructed to classify images. With the proliferation of digital cameras, scanners, and better image editing software people have become more interested in pictures than words. As the old saying goes, “a picture is worth a thousand words.” With the explosive growth of images on the Internet, ways to find and select images of a particular type need to be developed.

Future work on this project should:

- Add more features that can be extracted. There are many more papers that work to classify only two image types, and examine various features that satisfy that particular classification. These features should be added to create a wider range of features to select from.
- A nice graphical user interface should be constructed that allows the user to pick which features they want to examine.
- Fixing any bugs or inappropriate program behavior from uncaught exceptions and so forth.

- Add more ways to visualize data. Parallel coordinate plots are only one method of visualizing data. There are other more complex methods that may provide more valuable visual analysis to the user.

APPENDIX A

GRAPHICS, VISUALIZATION, AND IMAGING GROUP WEB SITE SOURCE CODE

APPENDIX B

NVIZ SUPPLEMENTAL DOCUMENTATION AND SELECTED SOURCE CODE

APPENDIX C

IMAGE FEATURE ANALYSIS SOURCE CODE