

Climate Data Visualization

Andrew Pellett

Advisors: Prof. Peter Koons, Sean Birkel

*Department of Electrical and Computer Engineering
University of Maine
Orono, USA*

andrew.pellett@umit.maine.edu

Abstract

This document describes the motivation for and design of a MATLAB Graphical User Interface (GUI) and the underlying program for generating visualizations of climate data. The program works with data in the netCDF format from the NCEP/NCAR reanalysis project to create the visualizations. The visualizations can be output as AVI video, still images, and still images with map projections applied. Integration with Google Earth is also available using the non-projected still images and the corresponding KML output file.

I. PREFACE

As this project progressed, development effort oscillated between improving the GUI, and improving the underlying program. Development began off of existing visualization code by University of Maine graduate student Sean Birkel. This document will discuss the development of this program categorically, rather than chronologically. This direction will avoid confusion from the jumping around between topics that occurred during the development.

II. GOALS

The goal of this project was to improve upon existing code for generating still images and KML given NCEP/NCAR reanalysis project data, and then to wrap it up in an easy-to-use GUI. This tool will potentially be useful for climate researchers and university students, as well as primary and secondary school students. The visualizations generated by this program could also be used in presentations at conferences as well as in lectures. In particular, this GUI could be used by middle school students in conjunction with the Maine laptop program to create a more interactive environment for earth science studies.

III. WHY MATLAB?

This project began with a choice of what language to develop in: C or MATLAB. C provides the fastest operation, but would be more challenging to develop in considering MATLABs included set of high level routines. MATLAB, while slower, comes with all sorts of high level visualizations tools,

⁰This project is co-funded by the ASSURE program of the Department of Defense in partnership with the National Science Foundation REU Site program under Grant No. 0754951. Thanks to Professor Peter Koons and Sean Birkel.

as well as tools made specifically for working with geospatial data (via the mapping toolbox). MATLAB code is also generally easier to modify, since the programmer doesn't have to worry about variable typing or memory allocation. A final advantage to using MATLAB is its existing GUI framework, which, while perhaps not as powerful as a standalone GUI framework like GTK, provides all the functionality necessary for this project. While MATLAB is slower than C, this would likely only be an issue in generating visualization sets that cover many frames and/or are at a particularly high resolution. Visualization sets such as these would most likely be preconceived and thus would be able to be prepared early, making the run time less of an issue. The MATLAB code is capable of producing previews of any frame within a couple seconds, which is quick enough for classroom use.

IV. NETCDF

The program operates on data in network Common Data Format (netCDF) from the National Centers for Environmental Prediction (NCEP) reanalysis project at the National Oceanic and Atmospheric Administration (NOAA) Earth System Research Laboratory (ESRL), Physical Sciences Division (PSD) [1].

The netCDF format is often used for storing scientific data. It is made to handle large amounts (gigabytes) of data, and there are a variety of tools for parsing, creating, and transferring netCDF files. A netCDF file has attributes, variables, and dimensions. Attributes are information about the data (metadata). Variables define the type and information about a certain set of data contained within the file (variables can have their own attributes). Variables also define how to access a specific variable, for instance:

```
short air(time, level, lat, lon) ;
```

This variable line, obtained using the ncdump utility, says that in order to get a single value of the air variable, the user must provide the corresponding time, level, latitude, and longitude, all of which are other variables. Dimensions define the number of values that exist for each variable. A netCDF file will have an arbitrary number of dimensions that are bounded in size, and one that is unbounded and can grow:

```

dimensions:
lon = 144 ;
lat = 73 ;
level = 17 ;
time = UNLIMITED ; // (1464 currently)

```

In this case, the time dimension is unlimited, which allows for data to be appended to the file over time without having to rewrite the whole file.

These particular netCDF files use the COARDS convention [2], which guarantees the existence of a certain set of file attributes and characteristics. This makes it possible for the processing code to grab something like the label for a colorbar from a file attribute, and have that label be correct across a span of different variables. The data in these netCDF files are packed:

$$packedValue = \frac{actualValue - offset}{scaleFactor}$$

This generally yields large, negative numbers. To get back to the actual data, the data is unpacked:

$$actualValue = packedValue * scaleFactor + offset$$

This allows the individual data points to be stored as short integers, reducing file size.

V. MODULARITY

The GUI and drawing methods of the program are independent of the incoming data, so given any set of X (longitude), Y (latitude), Z data, the program should be able to generate the same type of quality output, with some slight adjustments to the colormaps and color range. One possible use for this would be to replace the part of the code that gets the data from the netCDF file with code that takes data from another input and prepares it as proper XYZ data. This would allow the program to be ported to handle data from the University of Maine Ice Sheet Model (UMISM), among other data sets.

VI. KML OUTPUT

One of the exciting functionalities of this program is its ability to produce output that can take advantage of the power of Google Earth to produce three dimensional visualizations of the data overlayed onto the globe. Google Earth provides a robust platform for viewing the globe from different perspectives, as well as integrating animations over time. All the program has to do is give Google Earth some standard .png images to wrap around the globe, and Google Earth does the rest.

In order to interface with Google Earth, a KML (keyhole markup language) document must be created. The KML file tells Google Earth the who, what, where, and when for each image. For instance, the colorbar image (who) is a screen overlay (what) that should be placed in the upper-right-hand corner of the screen (where) indefinitely (when). Additionally, a single frame of the animation (what) is a ground overlay (what) that should be wrapped around the globe, given a set of starting coordinates for a corner of the image (where) for however long is specified in its TimeSpan tag (when). An example of how to construct a KML file for a basic Google Earth animation is included in the appendix. The documentation for KML is available at [3].

VII. CAPTURING IMAGES

The program uses two different methods to capture images, *print()* and *getframe()*, both of which are MATLAB built-in functions. It was confirmed using the MATLAB profiler that the image capture part of the program is by far the most time consuming, dwarfing even the generation of the image.

A. *print()*

The *print()* function captures a MATLAB figure in its current state at a certain resolution, given in Dots-Per-Inch (DPI). This method generates the highest quality images for all resolutions. The run time of the *print()* function increases significantly with the DPI. This method is used to capture the highest quality still images for KML and projected output.

Unfortunately, *print()* is capable of capturing only the whole figure, not just a child axes within a figure. Since uicontrols (GUI elements) are also children of the figure, they are also captured in the output image, which is not desired. There is a *print()* argument *-noui* which doesn't print the uicontrols, but it still prints the whitespace they occupy, which makes the argument useless in this application.

B. *getframe()*

The *getframe()* function captures a MATLAB graphics object in its current state, at its current resolution. The *getframe()* function is capable of capturing just an axes, not its parent figure, which is more useful for a GUI. The run time of the *getframe()* function is less than that of the *print()* function, creating a speed versus quality trade-off. This method is used to capture quicker, lower quality still images, as well as individual AVI video frames.

The *getframe()* function suffers from the fact that it captures the image at the exact current resolution. This means that, to a point, better images can be generated by increasing the window size or perhaps even using a larger monitor. The quality of the images captured by *getframe()* are acceptable at their native resolution, but in order to obtain images at higher resolutions, the image data has to be interpolated, which causes a loss in quality. The MATLAB function *interp2()* is used to interpolate the image to a user-input size. This

interpolation leaves a moderate amount of noticeable artifacts. It is possible that the use of better image processing methods (different interpolation method, filtering) could produce more acceptable quality images.

Still images captured with the *getframe()* method can be interpolated to any resolution. Alternatively, the user can lock the original aspect ratio of the image, input one dimension, and the program will automatically set the second dimension. This feature was added in an attempt to possibly improve image quality, but doesn't seem to reduce the visibility of artifacts.

VIII. OUTPUT MODES

The program facilitates three different modes of output: preview, KML still images, and AVI video. Each output mode has its own use and obeys its own user-input parameters.

Preview mode allows the user to view data visualizations in the GUI without actually writing them to files. This is potentially useful in a classroom environment where each student, or small groups of students, are assigned to look at a specific date in time to make observations. This method allows quick viewing without the overhead of writing a file, then opening it in an image viewer.

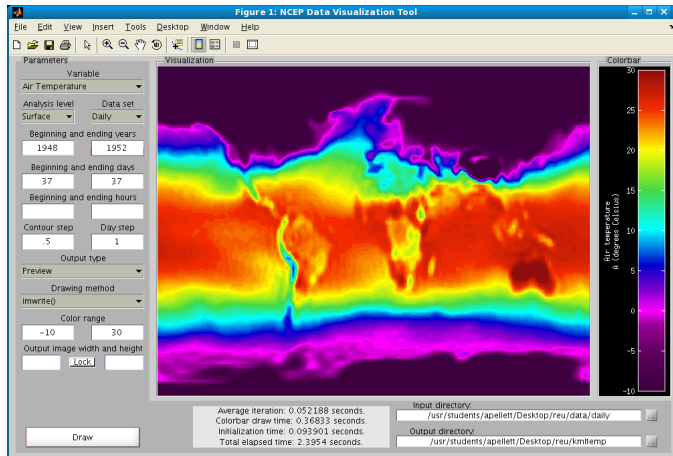


Fig. 1. Example of preview mode.

The KML output mode generates a KML file along with a series of still images which can be used by themselves, for instance, in presentations, or in conjunction with Google Earth to create a powerful three-dimensional visualization.

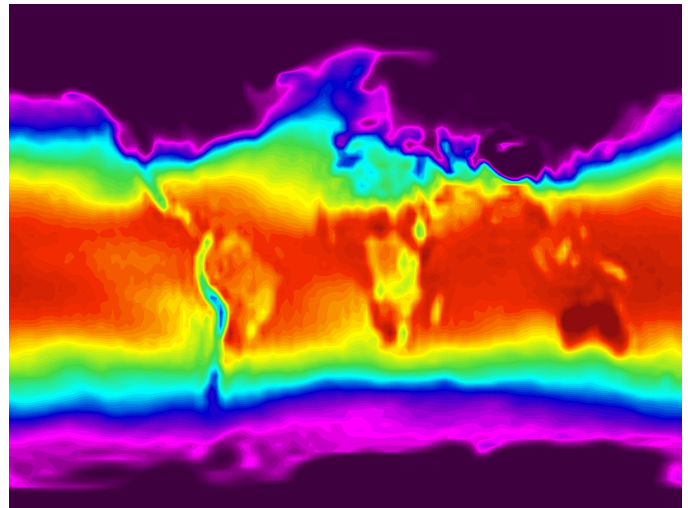


Fig. 2. Example of still image output

AVI video mode creates standalone animations, complete with colorbar. This allows for presentation of animations in an environment where Google Earth isn't available; for instance, on a web site.

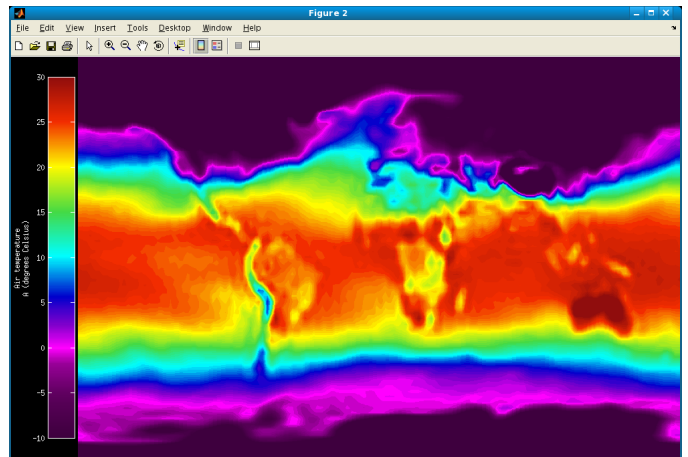


Fig. 3. Example of an AVI frame

IX. GUI DESIGN

MATLAB provides a framework for creating a graphical user interface; most of this functionality resides in the *uicontrol()* function. This function is capable of producing a variety of user interface elements, such as pop-up menus, buttons, check boxes, and editable text fields. Each user interface element (other than the static text elements) has a corresponding callback, which is a function that is called any time the UI element is modified. Generally, each callback will set a parameter or set of parameters that define what the program will do when the visualization process is initiated.

The GUI design for this project is primarily a basic one, mostly using stander uicontrols. There are three major

sections of the GUI: the uicontrols, the drawing axes, and the color bar axes. The uicontrols section is composed of two uipanel that hold all of the uicontrols. The drawing and color bar axes are simply children of the main figure. All positioning and sizing is done using normalized units, so the GUI scales as the window size changes, although drastic changes, especially downsizing, can cause the interface to become skewed. The native size is ideal as it is small enough to fit on one of the Maine laptop program laptops.

In MATLAB, there are three objects which can contain other objects: figures, uipanel, and axes. A figure is always the first container, and can't contain another figure. Uipanel can be children of figures, and can be parents of axes. Axes can be children of figures or uipanel, and are parents to graphics objects such as lines, points, and surfaces. Unfortunately, the *print()* function is only capable of capturing entire figures, so in order to capture higher quality still images, a separate figure window is opened, and the drawing axes has its parent changed to the new figure, then changed back to the original figure after the image is captured. For capturing AVI frames, a new, temporary figure window is also created, and both the drawing and colorbar axes have their parent changed. The colorbar is redrawn to the outside-left of the drawing axes, the frame is captured, and the colorbar and drawing axes have their parents changed back to the original figure.

Most of the controls used in this GUI are standard uicontrols, including the pop-up menus, editable text boxes, static text, toggle button, and draw button. For selecting input and output directory, the *uigetdir()* function is called, which provides a standard graphical file directory navigator for selecting the desired directory. The status text box that updates as each frame is drawn is implemented as four static text objects which have their String property set repetitively.

The status text box provides information about the most recently completed frame, including the data it's visualizing, the iteration, and the amount of time it took to generate the visualization. After all frames are drawn, average and overall statistics are displayed.

X. CONCLUSION

Over the course of this project a flexible, easy-to-use MATLAB application for the visualization of geological data from the NCEP/NCAR reanalysis project has been produced. The program is capable of taking user input through a graphical user interface, and producing visualizations in the form of AVI videos, projected still images, and still images with KML, ready for integration with Google Earth.

REFERENCES

- [1] (2008) Ncep/ncar reanalysis project. [Online]. Available: <http://www.cdc.noaa.gov/cdc/reanalysis/reanalysis.shtml>
- [2] (2008) Coards netcdf convention. [Online]. Available: http://ferret.wrc.noaa.gov/noaa_coop/coop_cdf_profile.html
- [3] (2008) Google kml documentation. [Online]. Available: <http://code.google.com/apis/kml/documentation/>

APPENDIX

A. Basic Animations using Google Earth and KML

Two overlay types:

- Ground overlay
 - Image is overlaid on the earth, and can be viewed from different perspectives by moving the camera.
 - `< groundoverlay > ... < /groundoverlay >`
- Screen overlay
 - Image is overlaid on the Google Earth window, and remains in the same position on the screen, independent of camera motion.
 - `< screenoverlay > ... < /screenoverlay >`

There are attributes which can be assigned to overlays. These attributes facilitate the placement of images on the globe, and the animation of those images.

Screen overlay attributes:

- Name
 - Gives a particular overlay a name to be displayed with the image
 - `< name > ... < /name >`
- Time span
 - Specifies how long an overlay should be displayed using two tags
 - * `< begin > yyyy-mm-ddThh:mm:ssZ < /begin >`
 - * `< end > yyyy-mm-ddThh:mm:ssZ < /end >`
 - `< timespan > ... < /timespan >` (timespan contains begin and end tags)
- Icon
 - Specify the image to overlay using a hyperlink reference
 - * `< href > /path/to/image.png < /href >`
 - `< icon > ... < /icon >` (icon contains href tag)
- Latitude/longitude box
 - Specifies the edge boundaries of the overlay using four tags
 - * `< north > ... < /north >`
 - * `< south > ... < /south >`
 - * `< east > ... < /east >`
 - * `< west > ... < /west >`
 - All boundaries are in decimal degree form
 - `< latlonbox > ... < /latlonbox >` (latlonbox contains north, south, east, and west tags)

Ground overlay attributes:

- Name
- Icon
- overlayxy
 - Specifies a point of the overlay image to use as an anchor point (4 sub-attributes)
 - * x
 - * y
 - * xunits
 - * yunits
 - The values of x and y depend

B. Example KML file

```
<?xml version="1.0" encoding="utf-8" ?>
<kml xmlns="http://earth.google.com/kml/2.2">
  <Folder>
    <name>NCEP Surface Air Temperature Averages 1948–2007</name>

    <ScreenOverlay>
      <name>Color Bar</name>
      <Icon>
        <href>colorbar.png</href>
      </Icon>
      <overlayXY x="0" y="1" xunits="fraction" yunits="fraction"/>
      <screenXY x="0" y="1" xunits="fraction" yunits="fraction"/>
      <rotationXY x="0" y="0" xunits="pixels" yunit="pixels"/>
      <size x="72" y="235" xunits="pixels" yunits="pixels"/>
    </ScreenOverlay>

    <GroundOverlay>
      <name>SAT 01–31</name>
      <TimeSpan>
        <begin>1948–01–01T00:00:00Z</begin>
        <end>1948–01–31T23:59:59Z</end>
      </TimeSpan>
      <Icon>
        <href>NCEP_air_01–31–1948–2007.png</href>
      </Icon>
      <LatLonBox>
        <north>90</north>
        <south>–90</south>
        <east>180</east>
        <west>–180</west>
      </LatLonBox>
    </GroundOverlay>

    <GroundOverlay>
      <name>SAT 02–29</name>
      <TimeSpan>
        <begin>1948–02–01T00:00:00Z</begin>
        <end>1948–02–29T23:59:59Z</end>
      </TimeSpan>
      <Icon>
        <href>NCEP_air_02–27–1948–2007.png</href>
      </Icon>
      <LatLonBox>
        <north>90</north>
        <south>–90</south>
        <east>180</east>
        <west>–180</west>
      </LatLonBox>
    </GroundOverlay>

    <GroundOverlay>
      <name>SAT 03–31</name>
      <TimeSpan>
        <begin>1948–03–01T00:00:00Z</begin>
```

```

        <end>1948-03-31T23:59:59Z</end>
    </TimeSpan>
    <Icon>
        <href>NCEP_air_03-29-1948-2007.png</href>
    </Icon>
    <LatLonBox>
        <north>90</north>
        <south>-90</south>
        <east>180</east>
        <west>-180</west>
    </LatLonBox>
</GroundOverlay>

<GroundOverlay>
    <name>SAT_04-30</name>
    <TimeSpan>
        <begin>1948-04-01T00:00:00Z</begin>
        <end>1948-04-30T23:59:59Z</end>
    </TimeSpan>
    <Icon>
        <href>NCEP_air_04-28-1948-2007.png</href>
    </Icon>
    <LatLonBox>
        <north>90</north>
        <south>-90</south>
        <east>180</east>
        <west>-180</west>
    </LatLonBox>
</GroundOverlay>

<GroundOverlay>
    <name>SAT_05-31</name>
    <TimeSpan>
        <begin>1948-05-01T00:00:00Z</begin>
        <end>1948-05-31T23:59:59Z</end>
    </TimeSpan>
    <Icon>
        <href>NCEP_air_05-31-1948-2007.png</href>
    </Icon>
    <LatLonBox>
        <north>90</north>
        <south>-90</south>
        <east>180</east>
        <west>-180</west>
    </LatLonBox>
</GroundOverlay>

<GroundOverlay>
    <name>SAT_06-30</name>
    <TimeSpan>
        <begin>1948-06-01T00:00:00Z</begin>
        <end>1948-06-30T23:59:59Z</end>
    </TimeSpan>
    <Icon>
        <href>NCEP_air_06-30-1948-2007.png</href>
    </Icon>

```

```
<LatLonBox>
  <north >90</north >
  <south >-90</south >
  <east >180</east >
  <west >-180</west >
</LatLonBox>
</GroundOverlay>

<GroundOverlay>
  <name>SAT 07-31</name>
  <TimeSpan>
    <begin >1948-07-01T00:00:00Z</begin >
    <end >1948-07-31T23:59:59Z</end >
  </TimeSpan>
  <Icon>
    <href >NCEP_air_07-30-1948-2007.png</href >
  </Icon>
  <LatLonBox>
    <north >90</north >
    <south >-90</south >
    <east >180</east >
    <west >-180</west >
  </LatLonBox>
</GroundOverlay>

<GroundOverlay>
  <name>SAT 08-31</name>
  <TimeSpan>
    <begin >1948-08-01T00:00:00Z</begin >
    <end >1948-08-31T23:59:59Z</end >
  </TimeSpan>
  <Icon>
    <href >NCEP_air_08-29-1948-2007.png</href >
  </Icon>
  <LatLonBox>
    <north >90</north >
    <south >-90</south >
    <east >180</east >
    <west >-180</west >
  </LatLonBox>
</GroundOverlay>

<GroundOverlay>
  <name>SAT 09-30</name>
  <TimeSpan>
    <begin >1948-09-01T00:00:00Z</begin >
    <end >1948-09-30T23:59:59Z</end >
  </TimeSpan>
  <Icon>
    <href >NCEP_air_09-28-1948-2007.png</href >
  </Icon>
  <LatLonBox>
    <north >90</north >
    <south >-90</south >
    <east >180</east >
    <west >-180</west >
```

```
        </LatLonBox>
    </GroundOverlay>

    <GroundOverlay>
        <name>SAT 10-31</name>
        <TimeSpan>
            <begin>1948-10-01T00:00:00Z</begin>
            <end>1948-10-31T23:59:59Z</end>
        </TimeSpan>
        <Icon>
            <href>NCEP_air_10-31-1948-2007.png</href>
        </Icon>
        <LatLonBox>
            <north>90</north>
            <south>-90</south>
            <east>180</east>
            <west>-180</west>
        </LatLonBox>
    </GroundOverlay>

    <GroundOverlay>
        <name>SAT 11-30</name>
        <TimeSpan>
            <begin>1948-11-01T00:00:00Z</begin>
            <end>1948-11-30T23:59:59Z</end>
        </TimeSpan>
        <Icon>
            <href>NCEP_air_11-30-1948-2007.png</href>
        </Icon>
        <LatLonBox>
            <north>90</north>
            <south>-90</south>
            <east>180</east>
            <west>-180</west>
        </LatLonBox>
    </GroundOverlay>

    <GroundOverlay>
        <name>SAT 12-31</name>
        <TimeSpan>
            <begin>1948-12-01T00:00:00Z</begin>
            <end>1948-12-31T23:59:59Z</end>
        </TimeSpan>
        <Icon>
            <href>NCEP_air_12-30-1948-2007.png</href>
        </Icon>
        <LatLonBox>
            <north>90</north>
            <south>-90</south>
            <east>180</east>
            <west>-180</west>
        </LatLonBox>
    </GroundOverlay>
</Folder>
</kml>
```