A Program for Handling Map Projections of Small-Scale Geospatial Raster Data

Michael P. Finn[a], Daniel R. Steinwand[b], Jason R. Trent[c,1], Robert A. Buehler[c,2], David M. Mattli[c], and Kristina H. Yamamoto[a]
[a]U.S. Geological Survey (USGS), Center of Excellence for Geospatial Information Science (CEGIS), Denver, Colorado
[b]USGS, Earth Resources Observation and Science Center, Sioux Falls, South Dakota
[c]USGS, CEGIS, Rolla, Missouri
mfinn@usgs.gov
[1]Now with Sandia National Laboratory, Albuquerque, NM.
[2]Now with Google, New York, NY.

ABSTRACT

Scientists routinely accomplish small-scale geospatial modeling using raster datasets of global extent. Such use often requires the projection of global raster datasets onto a map or the reprojection from a given map projection associated with a dataset. The distortion characteristics of these projection transformations can have significant effects on modeling results. Distortions associated with the reprojection of global data are generally greater than distortions associated with reprojections of larger-scale, localized areas. The accuracy of areas in projected raster datasets of global extent is dependent on spatial resolution. To address these problems of projection and the associated resampling that accompanies it, methods for framing the transformation space, direct point-to-point transformations rather than gridded transformation spaces, a solution to the wrap-around problem, and an approach to alternative resampling methods are presented. The implementations of these methods are provided in an open-source software package called MapImage (or *mapIMG*, for short), which is designed to function on a variety of computer architectures.

Note: Any use of trade, firm, or product names in this article is for descriptive purposes only, and does not imply endorsement by the US government.

INTRODUCTION

"Each method of map making has something peculiar to itself. When the earth is delineated on a sphere, it has a shape like its own, nor is there any need of altering at all. Yet it is not easy to provide space large enough (on a globe) for all of the details that are to be inscribed thereon; nor can one fix one's eye at the same time on the whole sphere, but one or the other must be moved, that is, the eye or the sphere, if one wishes to see other places."

– Claudius Ptolemy (90–168 A.D.)

Raster datasets with global or near-global coverage are now commonly available, in part due to advances in remote sensing technologies and computer storage and processing. These large datasets allow researchers from across the world to conduct small-scale geospatial modeling projects with easily comparable results. In addition, research on vast areas, such as across a continent, can be accomplished more quickly and efficiently when access to a global dataset is available. An inherent issue with these datasets is that the original projection may not be the most desirable to a researcher, and therefore reprojection becomes necessary; often times it is beneficial to combine multiple global data sets in different projections to a common projection base (Ilifee and Lott 2008).

While existing reprojection software works well at the local scale, some implementations produce unexpected and erroneous results when working with raster datasets with global coverage (Usery et al. 2003a; Wonders 2011). Usery and Seong (2001) investigated the effects of global map projections on the accuracy of calculated areas in raster datasets and showed them to be dependent on both spatial resolution and latitude, and, further, they found that no single projection was best for all uses. This paper reviews the processing procedures for such small-scale data, particularly as they relate to geometric issues. These issues include distortions that are often present in large area transformations, and the associated resampling issues. The paper also provides documentation for an open-source software, *mapIMG*, that can successfully reproject such large datasets.

Several issues that affect overall product quality need to be considered when reprojecting raster map data. Very generally, the more coarse the spatial resolution, the more prominent the problems can be, especially with regard to how various map projections contribute differently to problems based on their derivation (properties). All of the following factors can affect the reprojected data, and are therefore covered in this paper: coordinate framing of the resulting output space; selection of a forward or inverse mapping model; selection of a gridded (and interpolated) or point-by-point method of querying the mapping model; and choice of resampling method.

All map projections have distortions. At a more local level (across a few kilometers), transformations between two map projections tend towards the linear, but as one moves towards the continental or global scale, these transformations are highly non-linear; thus, the distortions resulting from reprojection of global raster data are much more of a concern than are local

area distortions (Steinwand et al. 1995). As a result, reprojecting global raster datasets—such as the continuous ASTER Global DEM (NASA 2011) or categorical global vegetation (Matthews 1983)—can produce accuracy errors in the reprojected data as a function of both scale and map transformation. In addition, the reprojection process can exceed the limits of traditional algorithms, causing software designed for large-scale, local area data to perform incorrectly (triggering an error condition) when used with datasets of global extent. Algorithms used for reprojecting global datasets need to be written specifically for this task.

In this paper, we discuss coordinate framing issues, show an algorithm used for the reprojection process and approaches to using it, examine the resampling process, and, finally, present the implementation of *mapIMG* to better handle these issues that are unique to raster datasets and that are not present when projecting point data. (For example, a cell is a projection of four corners and a resampling function must accompany it to populate the new cell value.) This program was used as the basis for other research that demonstrated distinctions between the new software and other software packages that a user might instead employ to reproject a small-scale raster dataset (Usery et al. 2003a and 2003b). In particular, Usery and others (2003a) showed that some software packages can introduce errors when projecting raster datasets, such as failing to use exact projection equations, and issues when handling data near the poles[1]. More recent work (Wonders 2011) shows that many of these problems still persist in some software packages. Comparisons between these software packages and *mapIMG* illustrated that the ability of *mapIMG* to check transformations in the forward direction, after the inverse transformations, reduces projection errors. Therefore, the benefits of *mapIMG* have already been tested in previous work. The goals of this paper are to expand upon the issues related to raster dataset reprojection and provide documentation highlighting how *mapIMG*'s implementation can more efficiently process small-scale dataset reprojections.

The goals of this paper are to expand upon the issues related to raster dataset reprojection and provide documentation highlighting how mapIMG's implementation can more efficiently process small-scale dataset reprojections.

I. MAP PROJECTION COORDINATE TRANSFORMATION PACKAGES

Map projection science is primarily concerned with the basic mathematics behind the theories and methods for mapping (Yang et al. 2000). Map projections provide an understanding of the mathematical relations of the spatial element on a map. This understanding can yield significant insight into informational properties within a given region (Bugayevskiy and Snyder 1995). The recent development and advancement of GIS and remote sensing platforms for managing and controlling digital data has brought about an explosion in the quantity of geospatial data. With this explosion has come an increasing utilization of spatial data, which are often in a map projection that is different from the one in which the data will ultimately be required (Ilifee and Lott 2008). Because of the above advances and problems, there is an increased need for automated projection transformations.

[1] These test datasets are available at http://cegis.usgs.gov/projection/acc_proj_data.html

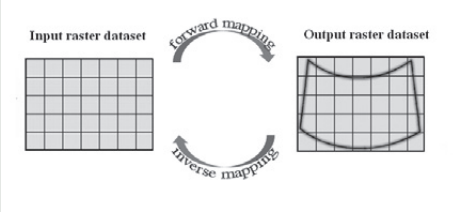

Figure 1. Workflow process for a typical automated raster projection transformation. The line, sample (row, column) coordinates from an input raster dataset are converted to projection coordinates in the projection system defined for the input raster dataset Likewise, the resulting output raster dataset dimensions are determined in both line, sample coordinates and in projection coordinates defined by the output projection system. These line, sample to projection coordinate mappings are typically simple first-order transformations. The reprojection process then takes place by either mapping from the output projection coordinates to the input projection coordinates (inverse mapping) or from the input projection coordinates to the output projection coordinates (forward mapping) by using the point-by-point map projection transformation package.

Software subroutine packages that perform map projection transformations usually operate on a point-by-point basis. The typical workflow process for such programs is outlined in Figure 1. The next few sections address projection transformation processes and problems for raster-specific automated projection transformations.

II. PROJECTION TRANSFORMATION PROCESS: COORDINATE FRAMING

The frame of a raster dataset defines the extent of the raster dataset in the projection space and also the alignment of projection space with the raster dataset coordinate system. Equations 1 through 4 define this relationship for projection grids that are aligned with line, sample (row, column) grids without rotation:

$$X = ULprojX + ((sample - 1) * pixelSizeX) \quad (1)$$

$$Y = ULprojY - ((line - 1) * pixelSizeY) \quad (2)$$

or, alternatively:

$$Sample = ((X - ULprojX) / PixelSizeX) + 1 \quad (3)$$

$$Line = ((ULprojY - Y) / pixelSizeY) + 1 \quad (4)$$

where, ULprojX is the upper-left X projection coordinate that corresponds to the upper-left-most sample in the raster dataset and ULprojY is the Upper Left Y projection coordinate that corresponds to the upper left-most line in the raster dataset. As a raster dataset coordinate pair, it is equal to (1,1). This is sometimes called "1-relative coordinates" because of this

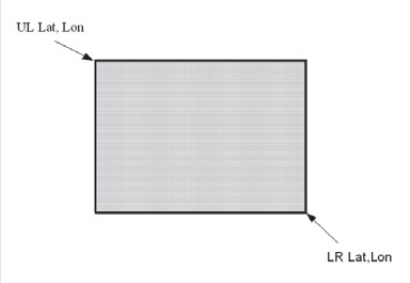

Figure 2. User defined output raster dataset extent as the upper-left (UL) and lower-right (LR) geographic coordinates.

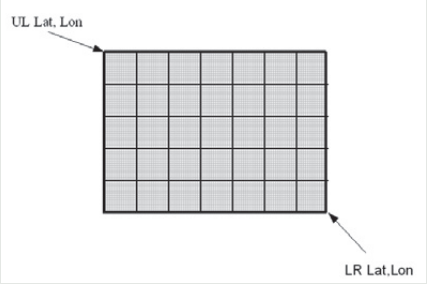

Figure 3. Conceptualized frame in geographic space of the input raster dataset, with grid lines added for clarification.

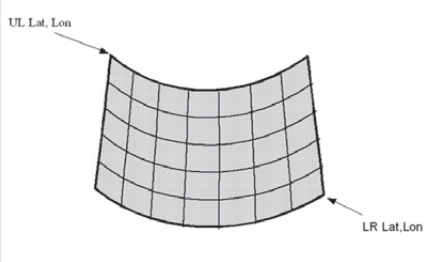

Figure 4. This space is converted to the output projection frame. Corners and sides of the frame are converted (piecewise) and projection coordinate minimums and maximums are recorded.

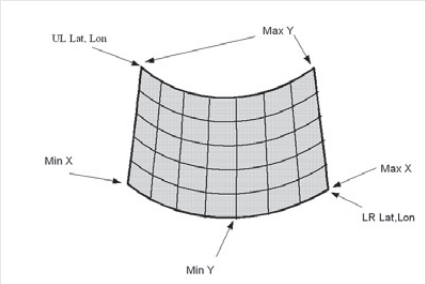

Figure 5. Locations of the minimum/maximum projection coordinates are noted. Maximums are recorded.

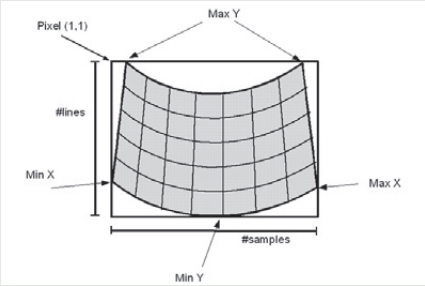

Figure 6. The minimum and maximum extents form the MinBox—this is the extent of the output raster dataset frame. The number of lines and samples are determined by dividing these dimensions by the pixel size.

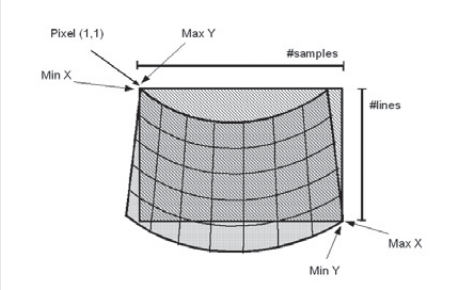

Figure 7. If the minBox algorithm is not applied, and only the UL and LR geographic coordinates are used to determine projection min/max, clipping of the frame can occur.

relationship to the UL (upper-left coordinate) = (1,1). (Note that different packages will define this somewhat differently; some are 0-relative, some define it at 0.5, 0.5, etc.)

This first-order transformation translates and scales coordinates and also defines the relationship between the raster (line, sample) grid and the dataset's projection grid. There is usually no rotation between the two coordinate systems. If there were a rotation, the above equations would need to be expanded to handle that condition. A common practice is to reference the upper-left raster dataset pixel to the projection coordinate system by specifying the projection coordinate of the center (or, just as commonly, the upper-left corner) of that pixel in the raster dataset.

DETERMINATION OF THE OUTPUT FRAME

The first step in changing the projection of a raster dataset to a new projection is to determine the geographic extent of the output space. We refer to this as the "output frame" and it is specified in the units of the output projection. The projection coordinates of the output raster dataset correspond in a linear fashion to the raster dataset coordinates of the output raster dataset, as given in the transformation equations (1)–(4). Two common methods used to determine the output raster dataset frame are the "geographic minBox" and the "direct specification of output projection extent."

THE GEOGRAPHIC MINBOX

The geographic minBox is defined as the area of a box formed by the upper-left and lower-right geographic coordinates covering the user's area of interest (Figures 2 and 3) in the input raster dataset. The output raster dataset frame corresponds to the geographic minBox. The geographic minBox can be viewed as a rectangle drawn on a map in a Plate Carrée or Equirectangular projection, which completely covers the user's area of interest (AOI) up to, and including, the entire map in the input raster dataset. This rectangular area is projected onto the output projection coordinate system (Figure 4). This is usually implemented with an incremental stepping through the coordinates, with the step size sufficiently small, keeping track of the minimum and maximum projection coordinates along the way (Figures 4 and 5). This minBox operation defines the minimum and maximum projection coordinates of the output raster dataset frame (Figure 6). If only the corners of the geographic area of interest are used, then they become the minimum and maximum projection coordinates of the output raster dataset frame, although this may result in a clipping of the AOI (Figure 7).

DIRECT SPECIFICATION OF OUTPUT PROJECTION EXTENT

In this method, the user directly specifies the minimum and maximum projection coordinates of the output space. A modification of this method is to specify the upper left corner (or some other point of reference) of the raster dataset in projection coordinates and the number of lines and samples. In contrast to the geographic minBox, the direct specification of output projection extent does not rely on an algorithm to determine projection coordinate extents.

III. PROJECTION TRANSFORMATION PROCESS: FORWARD AND INVERSE MAPPING

After the output frame is determined, the pixels in the output raster dataset need to be populated with the appropriate values from the input raster dataset as determined by the projection transformation model. This transformation model is implemented using either a forward mapping approach or an inverse mapping approach—or a combination of both. Once the locations are determined, the process of resampling—that is, pixel interpolation, is applied to determine a final data value for the given pixel.

Forward transformation models would typically step along the input raster dataset pixels, converting each line, sample coordinate pair to projection coordinates in the input projection space, and—using the map projection transformation software on a point by point basis—map to output space projection coordinates and then finally to output raster dataset line, sample coordinates. An inverse model does the opposite: it steps along the output space line, sample grid that is being created, converts each raster dataset coordinate to a projection coordinate in the output space, uses the map projection transformation software to find the corresponding input projection coordinate, and converts it to the input space line, sample coordinate. The location of the point (projection coordinates (X, Y) in the input raster projection system) within a pixel used in the map projection transformation step is determined during the raster dataset framing process defined earlier; this defines how the projection grid aligns with the raster dataset (line, sample) grid, whether it be center-referenced, corner-referenced, or tied to some other reference point.

Because these map projection transformations are often not linear over the entire raster dataset space, results of the above transformations will not result in integer pixel locations—they will usually lie in between pixel postings. Therefore, some method of resampling is necessary to determine a pixel value in the output raster dataset. Oftentimes the nearest value is assigned (the nearest-neighbor method), but other methods are typically used for remote sensing datasets that rely on the signal characteristics of the cells and recognize contributions to the signal from neighboring pixels. Examples of these methods of resampling are bilinear interpolation (2x2 neighborhood) and cubic convolution (4x4 neighborhood). These neighborhood based interpolation resampling methods are typically not used with categorical data.

APPROACHES AND DETAILS ABOUT THE INVERSE MAPPING PROCESS

The inverse mapping algorithm for performing a projection conversion on a raster dataset works as follows:

For each line in the output raster dataset:

For each pixel in this output raster dataset line:

- Determine the output projection coordinate for this pixel using equations (1) and (2).
- Convert the coordinate to the input projection using a projection transformation package.
- Determine the input raster dataset coordinate using equations (3) and (4).
- Convert to the nearest pixel by adding 0.5 to the line and to the sample coordinate and then truncate and read the raster dataset value(s) at this input raster dataset coordinate.

The simple algorithm steps outlined above describe a point-by-point, nearest neighbor approach. This point-by-point process can be time-consuming when the number of pixels to process is large. The approach of several existing algorithms use grids, which can save a considerable amount of processing time. The gridded approach processes blocks of the raster dataset when the transformation between the two spaces is defined in an incremental linear fashion in two dimensions. Some local-area projection changes can be modeled in this fashion because at their large scale the geographic coordinates approximate a straight line with grid blocks. On the other hand, most projection transformations of data with global extent cannot be described in this incremental linear fashion because their smaller scale causes their geographic coordinates to be curvilinear.

The algorithm for *mapIMG* uses the point-by-point method. While some processing efficiencies can be gained by utilizing some incremental linear approximations where they are valid, we chose to implement an algorithm in the more rigorous point-by-point method.

At times, a potential problem remains after the completion of populating all values through the inverse mapping/resampling process. This problem is often referred to as the wrap-around problem; handling it is discussed next.

IV. PROJECTION TRANSFORMATION POTENTIAL PROBLEM: THE WRAP-AROUND PROBLEM

A problem referred to as "wrap-around" is not uncommon when projecting raster data on a global scale (Figure 8). The figure shows the result of projecting global land cover data from geographic coordinates to a Hammer IV projection, which is an equal-area projection with curved parallels, without concern for the appropriate geometry associated with projecting raster data of global extent. (This reprojection shown in Figure 8 could have been one of many different projections.) Note the replicated areas of Alaska and Siberian Russia on both sides of the map.

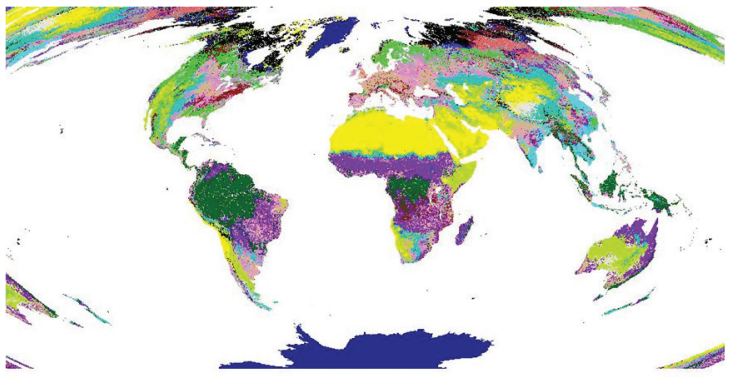

Figure 8. The wrap-around problem. Note the inapt geometry associated with projecting raster data of global extent.

The wrap-around problem shown in Figure 8 is an artifact arising from the use of inverse mapping. While inverse mapping has the computational advantage of only computing those pixels needed for the output raster dataset, it does pose problems when attempting to map a location that is located in the output raster dataset fill areas—that is, areas in which the projection transformation into the input space is not truly defined. In Figure 8, the Hammer IV projection results in an oval-shaped map of the globe. When we attempt to inverse map a point (pixel) in an area of fill (for example pixel (1,1)), the transformation between the line, sample space to the Hammer IV projection grid is well defined (it is a linear grid). However, when that resulting projection coordinate is placed in the map projection transformation software, one of two things could happen: it may cause an error condition, or it might be mapped to an incorrect location with valid data and without error conditions (often due to the periodic nature of trigonometric functions used in map projection transformations). This final condition is what is referred to as "wrap-around" and results in the effect seen in Figure 8. A simple but computationally expensive solution to this problem in the general case is to perform the inverse projection transformation and then perform the forward transformation on the result back into the output space to see if the output space coordinates match. If they do not, the condition is flagged as "wrap-around" and the output pixel is given a fill value (see Figure 9).

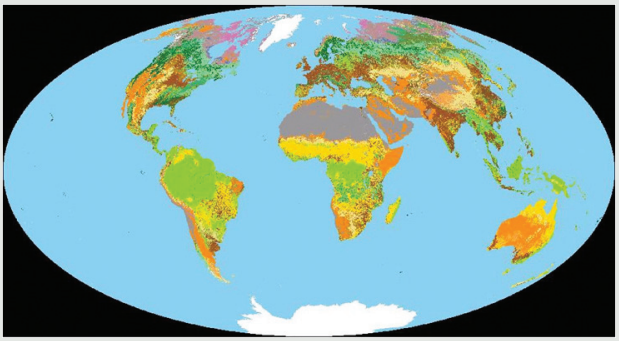

Figure 9. Projection of global land cover data to the Hammer IV projection. Note "wrap-around" problem fixed using fill value (from Steinwand 2003).

V. PROJECTION TRANSFORMATION POTENTIAL ISSUE: RESAMPLING

Kimerling (2002) outlined the spatial and mathematical nature of data loss and duplication during raster reprojection and resampling for three equal-area world map projections, showing explicitly the extent of loss and duplication at five-minute intervals using the ETOPO5 dataset (a continuous dataset). The choice of resampling methods deserves special attention when working with raster data of global or continental extent. Issues of concern are (a) the greater geometric distortions that are often present in large area map projection changes, such as a square cell being transformed to a parallelogram or a pixel to an entire raster dataset line (the pole in Mollweide converted to Equirectangular); and (b) the resampling that has to occur with signal-based continuous vs. categorical data. The errors from resampling in areas of large geometric distortion or scale change caused by a change in projection for categorical data (but not continuous/signal data) have been addressed by Steinwand (2003) and are reviewed below.

In general, projection transformation software packages are designed to work with points. Despite their point-specific approach, automated map projection software programs are used on both vector and raster data. Normally, a master routine provides the data in point form and keeps track of these point locations in pixels or vectors. In other words, they treat the raster data as just a grid of points and not as cells. In converse, software packages written specifically for raster datasets preserve (or should preserve) area (cell) relationships; more specifically, the (preserved) relationship is in the manner in which the transformation model uses the projection software, and then how the resampling over a cell is performed.

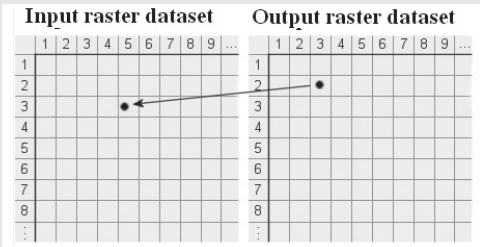

Figure 10. Nearest-neighbor mapping of one pixel in raster dataset (line, sample) space. Because the input and output are in different projection spaces, the corresponding raster dataset (line, sample) spaces will not be identical. This inverse mapping looks to the value in the input raster dataset (that the arrow points to) and uses that value to populate the pixel in the output raster dataset (from Steinwand 2003).

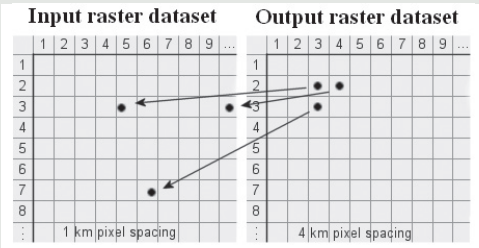

Figure 11. Nearest neighbor mapping of adjacent pixels when output resolution is reduced (from Steinwand 2003).

The nearest-neighbor resampling algorithm is used to determine an output pixel value by rounding to the nearest integer pixel location in the input raster dataset (Figure 10). This is because the resulting input raster dataset coordinates from the reprojection process are usually not at exact pixel center locations but are somewhere within that pixel. That pixel's value is assigned to the output raster dataset at the coordinate under study. Although this method is computationally efficient, it can result in a raster dataset that is not completely representative of the original data due to this within-pixel location that is not the center.

Occasionally, the next pixel in the output raster dataset, again mapped with the same algorithm, falls more than one pixel away in the input array (Figure 11). This can occur when the spatial resolution of the

output raster dataset is reduced, or as a result of the projection change transformation. The output raster dataset is then said to be under-sampled in that area.

Under these conditions (for example, when mapping a 1 km pixel to a desired 4 km pixel output), the output raster dataset does not include all data in the input raster dataset. More importantly, the nearest-neighbor resampling algorithm does not necessarily select a pixel that is representative of the area being resampled, but instead chooses the nearest one. This can result in areas of a raster dataset not being representative of the input raster dataset area, if, for example, a minority class happened to be the nearest pixel. For example, Figure 11 shows 1 km pixels that are spaced 4 km apart in the output raster dataset, instead of a pixel that truly represents the area of the 4 km pixel. Further, Figure 11 shows that the output pixel values (2,3) and (2,4) contain values from only the pixel values of (3, 5) and (3, 10) without capturing in any way the value between (3, 6) and (3, 9) horizontally.

Steinwand (2003) presented a new resampling algorithm for categorical data that addresses issues discussed in this section (Section V). The algorithm maps pixels as polygons rather than as points. As shown in Figure 12, the corners a, b, c, and d of the output pixel at sample 3, line 2, map to input locations A, B, C, and D. From this figure, one sees that 33 pixels, some of which are partial, comprise the data that could be considered for the output pixel. Once the input array pixels that are a part of the output raster dataset pixel footprint are determined, simple statistical methods or common resampling techniques can be applied to determine the output array pixel value that is to be assigned to the output array. For example, statistical methods such as the maximum occurring pixel, the minimum, and the mode, can be utilized for categorical data (Figure 13). Methods of resampling that are more complex—for example, that favor certain classes or that combine classes into composite classes—can also be used, at the cost of runtime performance.

VI. MAPIMAGE (MAPIMG): AN IMPLEMENTATION

The *mapIMG* program has been successfully used to investigate problems with raster dataset reprojection, particularly with global data (Usery et al. 2003a and 2003b). The program is able to handle raster datasets with varying spatial resolution and projections. This section describes the implementation of the software package covering the resampling option, the wrap-around problem, data manipulation, schematic and flow charts that show the relationship between the code and the user interface, the multiplatform capabilities, the major classes (programming constructs) used, and the two major functions of the program.

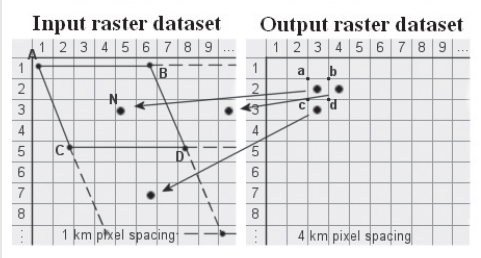

Figure 12. Mapping the input pixel's footprint—the concept behind the new algorithm. N is the center (or nearest-neighbor) point of the pixel being populated (from Steinwand 2003).

	1	2	3	4	5	6	7	8
1	21	71	52	23	31	22	71	95
2	95	21	52	31	22	24	23	71
3	52	95	71	71	23	52	31	95
4	52	71	24	21	31	52	95	52
5	71	95	24	22	71	95	31	24

Figure 13. Input raster dataset from Figure 12 with pixel values. Using these categorical values, the value of N from Figure 12 can be 31 (median), 21 (minimum), 95 (maximum), or 33 (mode).

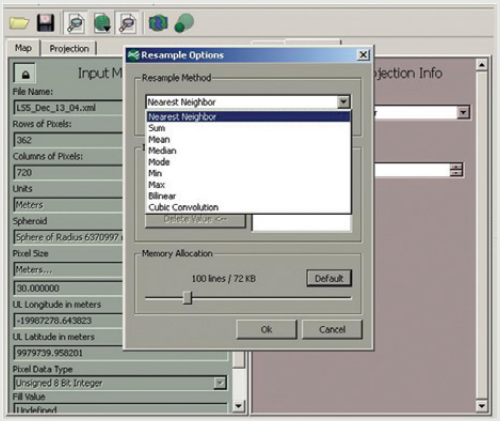

Figure 14. User options for resampling

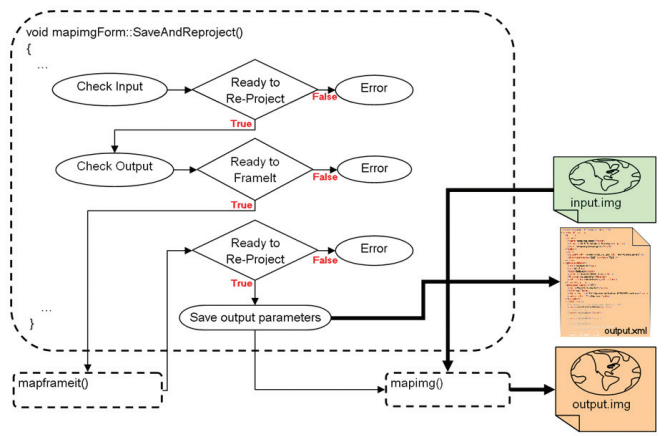

Figure 15. Schematic of mapIMG *program*

The user can select the sampling method they prefer (Figure 14). In addition, the program implements the inverse mapping process described previously and addresses the wrap-around issue. Figure 9 demonstrates a typical output product from the program (in this case, projection of global land cover data to the Hammer IV projection). When compared with Figure 8, it is noted that the wrap-around problem has been resolved by the *mapIMG* software. As a stand-alone program, *mapIMG* is available for various operating systems including MS Windows, UNIX (many variants), and Linux. It has a dialog box as a user interface and a menu bar and toolbar that appears upon execution of the program described in the User's Guide for this program (Finn and Mattli 2012).

The *mapIMG* program utilizes multiple computer input/output (I/O) techniques for manipulating data files. The program uses generic binary raster dataset, which are files with a value for each cell, sorted in a row major order without any header information. All raster dataset files must be accompanied by a metadata-type file describing the information about the raster dataset file contents (like the information contained in a header) for both the input and the output. Earlier versions of the *mapIMG* program use an ASCII file known as the .info files for this purpose. The current version of the program uses an .xml file for the same purpose. The .xml file is an easily read format that can assist in the importation of the generic binary file into other GIS packages.

Schematic and flow charts are shown in Figures 15 and 16, respectively. Figure 15 displays the relationship between the code for the interface, the two primary functions, and the input and output. Figure 16 details the logical flow of the process.

The *mapIMG* program is "multiplatform" and, thus, a researcher can port it to virtually any operating system. Users can download pre-compiled versions for Windows, Linux with K Desktop Environment 4, and UNIX for x86 and Sparc (Solaris 8 and 9) from the US Geological Survey (USGS) website (http://cegis.usgs.gov/projection/acc_proj_data.html). The source code is also available at the same site. A user can download and compile it for their own computing environment if they have the following three components: Nokia Qt (2012) installed the Tagged

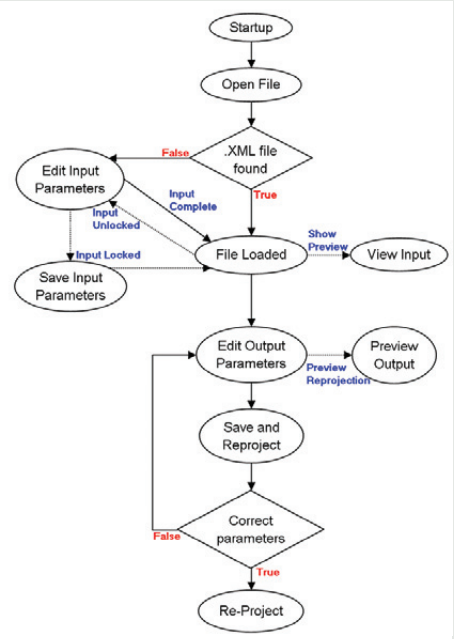

Figure 16. Diagram of reprojection flow

Image File Format (TIFF) library, and a C++ compiler. The TIFF library is used to convert incoming TIFF files to generic binary images (IMGs) which are then processed internal to the program.

Beyond the classes created in forming the GUI, which are transparent to the user, four major classes are used: RasterInfo, ResampleInfo, IMGINFO, and ImgIO<>. (Again, these classes are programming constructs, and not to be confused with classes of map projections.) These files are created by the code and do not have to be supplied by the user. RasterInfo is an encapsulation of all uses of the metadata. It stores the parameters as private attributes and it is used for passing the data among functions. When passed a filename it will automatically look for a corresponding .xml file. If none is found, it will look for the older .info file and replace it with an .xml file (Figure 15). (The .info file was used in an earlier version of the program. It functioned as a parameter file for input and output.) By checking for both the .xml file and the .info file, backward compatibility can be maintained. ResampleInfo stores all the parameters associated with the resampling method options. IMGINFO is a "struct" that holds a copy of the parameters needed in the GCTP function. The *mapIMG* program begins by copying information from the RasterInfo into the IMGINFO. While this may seem redundant, it actually helps to improve performance because it is faster to reference directly a public attribute than to use a get function. Finally, ImgIO<> is a templated class that encapsulates file I/O to the generic binary file as well as calls the GCTP function. It stores the input file in memory by lines using a "least recently used" cache. The larger the cache, the less often it will have to access the hard drive and request a read. This has helped to speed up greatly the operation of many reprojections of global raster datasets (Przybylski, 1990).

There are two major functions executed within the *mapIMG* program that work with GCTP: mapframeit() and mapimg(). The mapframeit () function is used to calculate the row and column dimensions as well as the coordinates of the upper left corner in a given projection. It begins by setting the minimum and maximum coordinates for each of the axes (pxmin, pxmax, pymin, and pymax) to be equal to their negative expected geographic coordinates of the selected map projection (pxmin = 180; pxmax = -180; pymin = 90; pymax = -90). It then loops through every latitude and longitude point with a grid resolution of 0.36 degrees of arc, or 500 rows by 1000 columns, and compares these points with the current minimum and maximum values of the output projection coordinates to determine if a point's values are outside those values—i.e., less than minimum or greater than maximum. If a value is outside, then it saves its value over the current one (Figures 6 and 7). Thus, at the completion of the algorithm, the output raster dataset space is defined.

After mapframeit() completes, the mapimg() function loops through every row and column (line and sample) in the output raster dataset and loads it with the appropriate values from the input raster dataset. If GCTP returns an error for a current point or detects the wrap-around condition then a "fill value" is placed there, otherwise, based on the resampling scheme, GCTP determines the input pixel that maps to the current output pixel and mapimg() loads the value from the input raster dataset. The mapimg() uses an additional code to check

the transformation in the forward direction after the inverse transformation to look for the wrap-around problem. This problem is evaluated by transforming a point to the old projection and then transforming that point back again. If the final point is different from the initial point then it is considered a wrap-around point. For time optimization, a row buffer is used in mapimg() to reduce I/O time by only writing to the output when an entire row has been populated.

CONCLUSIONS

Projection of global raster data can introduce significant errors unless proper reprojection techniques are utilized. The methods discussed in this paper minimize these errors by, first, framing the raster dataset, as defined by the extent of the raster dataset in the projection space. This extent is specified in the units of the output raster dataset map projection system, and is accomplished using a geographic minBox (the area contained in a box formed by the upper-left and lower right geographic coordinates covering the user's area of interest) and the specification of the output extent (the minimum and maximum projection coordinates of the output space). Second, the inverse mapping algorithm is applied by stepping through the output raster dataset space pixel-by-pixel, calculating the corresponding coordinates in the input raster dataset as it executes. By utilizing categorical resampling with modal categories, better results can be achieved than nearest-neighbor methods (based on desired outputs/visualization) when large changes in scale occur.

Although the benefits to *mapIMG* have been documented, there are potential limitations to the program. For example, in the cases of more complicated topology than those outlined previously, edge effects that affect the resampling operation process may be present. In addition, the geographic minBox may also encounter problems with unusual or complicated topology or arbitrary spherical rotations. Future studies include a more detailed analysis on comparing spherical versus ellipsoidal projections, evaluating the accuracy of multiple forward and inverse projections, and issues associated with aliasing due to resampling.

Because some software packages can introduce artifacts (the "wrap-around problem") when projecting raster datasets, *mapIMG* checks transformations in the forward direction after the inverse transformations to eliminate these reprojection artifacts. This is one of the primary benefits of *mapIMG* ; this paper expands upon issues related to raster dataset reprojection and provides documentation to highlight how *mapIMG*'s implementation can more efficiently process small-scale dataset reprojections. We presented the implementation of *mapIMG* to better handle issues that are unique to these raster datasets. The *mapIMG* program uses methods to provide solutions to the problems for raster data reprojection on a variety of computer architectures, including the often-overlooked problems of wrap-around and coordinate framing, especially for small-scale data. The program is freely available in both source code and executable forms.

ACKNOWLEDGMENT

The authors wish to thank Dr. E. Lynn Usery, USGS, for his scientific guidance to this project, and Dr. Tom Shoberg for helpful comments on the manuscript.

REFERENCES

Bugayevskiy, L. M., and J. P. Snyder. 1995. *Map Projections: A Reference Manual*. London: Taylor & Francis.

Finn, Michael P., and David M. Mattli. 2012. "User's Guide for the mapIMG 3: Map Image Reprojection Software Package." *US Geological Survey Open-File Report 2011–1306*, 12 p.

Ilifee, J. and R. Lott. 2008. *Datums and Map Projections: For Remote Sensing, GIS and Surveying.* 2nd Edition. Scotland, UK: Whittles Publishing.

Kimerling, A. J. 2002. "Predicting Data Loss and Duplication when Resampling from Equal-Area Grids." *Cartography and Geographic Information Science.* 29(2):111–126.

Matthews, E. 1983. "Global Vegetation and Land Use: New High-Resolution Data Bases for Climate Studies." *Journal of Climate and Applied Meteorology*, 22, 474–487.

NASA. 2011. ASTER: *Advanced Spaceborne Thermal Emission and Reflection Radiometer.* Internet at **http://asterweb.jpl.nasa.gov/index.asp**. Last accessed 10 July 2012.

Nokia. 2012. Qt - *A cross-platform application and UI framework.* Internet at **http://qt.nokia.com/**. Last accessed 10 July 2012.

Przybylski, S. A. 1990. *Cache and Memory Hierarchy Design: A Performance-Directed Approach.* Morgan Kaufmann Publishers Inc., San Francisco, CA.

Ptolemy, C. 90–168 A.D. *Geographia* (Chapter XX [Of the lack of symmetry in the map that accompanies the geography of Marinus], Book One). Quoted from Page 39 of Stevenson, E. L. 1932. *Geography of Claudius Ptolemy. Translator and Editor.* New York: The New York Public Library.

Steinwand, D. R. 2003. "A new approach to categorical resampling." *Proceedings of the American Congress on Surveying and Mapping Spring Conference,* Phoenix, AZ. Gaithersburg, MD: ACSM.

Steinwand, D.R., J. A. Hutchinson, and J. P. Snyder. 1995. "Map projections for global and continental data sets and an analysis of pixel distortion caused by reprojection. "*Photogrammetric Engineering and Remote Sensing,* v. 61, no. 12, December, pp. 1487–1497. Falls Church, VA: American Society for Photogrammetry and Remote Sensing.

Usery, E. L., M. P. Finn, J. D. Cox, T. Beard, S. Ruhl, and M. Bearden. 2003a. "Projecting Global Datasets to Achieve Equal Areas." *Cartography and Geographic Information Science*, v. 30, no. 1, pp. 69–79.

Usery, E. L., and J. C. Seong. 2001. "All Equal-Area Map Projections are Created Equal, but some are More Equal than others." *Cartography and Geographic Information Science.* 28(3):183–193.

Usery, E. L., J. C. Seong, D. R. Steinwand, and M. P. Finn. 2003b. "Accurate Projection of Small-Scale Raster Datasets." *Proceedings, 21st International Cartographic Conference, Durban, South Africa*. International Cartographic Association.

Wonders, B. 2011. "An Analysis of Introduced Projection Error from Commercial Software Packages." Proceedings, Association of American Geographers Annual Meeting, Seattle, Washington.

Yang, Q., J. P. Snyder, and W. R. Tobler. 2000. Map Projection Transformation: *Principles and Applications.* London: Taylor & Francis:.