Technical Report: alg02-001
October 11,2002

Tyler Alumbaugh, talumbau@ncsa.uiuc.edu
Peter Bajcsy, pbajcsy@ncsa.uiuc.edu

Automated Learning Group
National Center for Supercomputing Applications
605 East Springfield Avenue, Champaign, IL 61820

Georeferencing Maps with Contours in I2K

Abstract

In this report we present an overview of the georeferencing capabilities in our Image to Knowledge (I2K) software that enables registration of digital maps (raster data) with contours (vector data). This document outlines the theory and standards used for proper geographic referencing of digital maps. Given 2D maps and 3D contours, georeferencing transformations are described for translating between spatial points specified in three different coordinate systems: latitude/longitude, UTM, and pixel. In the I2K work, we interpret and utilize georeferencing parameter information embedded in different types of digital maps to provide a system that can geo-register contours for several popular map types. This document serves as a guideline for other scientists trying to geo-register maps and contours.

1. Introduction

Georeferencing, or geographic referencing, is the name given to the process of assigning values of latitude and longitude to features on a map. Latitude (lat) and longitude (long) describe points in three-dimensional (3D) space, while maps are inherently two-dimensional (2D) representations. With the advent of computers, modern maps are usually stored as digital images since they represent raster information similar to 2D image information. The steps involved in georeferencing a digital map image can vary among map image specification types, but the end result is the ability to retrieve the lat/long coordinates for any point on the georeferenced map. This capability is useful because the lat/long coordinates precisely define the position of an object on the Earth.

The reader will note that the georeferencing process involves *transformations* that occur for a variety of reasons. Some transformations are related to different file formats, others to different metadata representations, some to resolution, and others to the translations between various 2D and 3D coordinate systems. The translation between coordinate systems is the only transformation dictated by the problem domain. The others, while very important, are reflections of the lack of a single standard for specifying data in the domain.

Lat/long coordinates define a point on a 3D model of the Earth, while map coordinates represent a pixel – a row and column location on a 2D grid obtained from projecting some 3D model of the Earth onto a plane. 2D maps are easy to display and facilitate distance measurement, while 3D coordinates are accurate but cumbersome and have no standard length for different degrees of latitude and longitude. The best way to define the position of an object on a map is with relative horizontal (column) and vertical (row) distances: "Three kilometers north of object A and two kilometers west of object B". In contrast, the best way to specify the position on a 3D sphere is with relative angular offsets: "Five degrees north and six degrees west of A".

In general, georeferencing involves transformations of 3D (lat/long) coordinates to and from 2D (map) coordinates. Georeferencing transformations must reflect the geometric properties of the earth model and projection plane, and also support a mechanism for expressing distances and relative positional information across different coordinate systems.

Given 2D and 3D coordinate systems, there are three possible classes of transformations. First, those that go from 2D-to-2D, bringing into alignment points that represent identical real-world locations. An example of this is multi-modal raster data fusion, for instance, the fusing of synthetic aperture radar (SAR) images with electro-optical (EO) or infrared (IR) images. The second class of transformations, 3D-to-3D, is used for information integration such as the synthesis of road network and river information. The third class of transformations operates between 2D and 3D coordinates. Identifying the location of water wells defined in 3D lat/long coordinates on a 2D digital terrain map is an application example in this class of transformations. The transformations across coordinate systems of different dimensions, where one of the coordinate systems represents the Earth, are also called geo-registration. In our I2K work, we focus on the third class of transformations, those that support georeferencing maps (2-D raster data) with contours or boundaries of regions (3-D vector data).

Fields that apply georeferencing include geology, hydrology, atmospheric science, archeology, earthquake engineering, forestry, environmental engineering, water quality research, ecological research, military operations and territorial insurance. For example, the petroleum industry needs to know precise drilling locations, so accurate georeferenced maps are very important. The United States Geological Survey (USGS) and the Defense Mapping Agency both create and make extensive use of georeferenced images, many of which are available to the private sector.

Users of georeferencing information frequently exchange data and therefore have adopted some georeferencing terminology and Earth modeling standards. However, not all georeferenced maps follow the same standards, so identical information content can be represented using different data formats or map types. The I2K tools are designed to work seamlessly with several of the most popular standards.

In this document we outline the theory and standards used for proper geographic referencing of digital images. The main goal of the document is to provide a description of Image to Knowledge (I2K) georeferencing functionality. In Section 2 we describe other GIS software packages and some basic georeferencing principles. Individual georeferencing transformations and sources of necessary georeferencing information are presented in Sections 3 and 4. Section 5 covers specific details of georeferencing in I2K and Section 6 presents a summary and discussion of future work. Appendix A outlines one common map projection, and Appendix B presents equations used in the georeferencing transformations.

2. Current Solutions

A software package that can accept geospecific data and provide geographic referencing is called a *Geographic Information System* (GIS). One example of a GIS system is ArcGIS from ESRI [2], with components that include ArcView and ArcExplorer. ENVI, the Environment for Visualizing Images from Research Systems, Inc. [6], and a suite of applications known as the USGS Mapping Science Software from the U.S. Geological Survey [3] are other GIS systems.

There is nearly limitless information available describing georeferencing (see [1], [12]), but the level of understanding necessary to correctly georeference a single image can be rather daunting. The difficulty is due in part to the complications of projecting a three-dimensional surface, the Earth, onto a two dimensional object, a map. Understanding something about this process goes a long way towards understanding why certain parameters are needed to geographically orient a digital image.

All GIS systems require a certain amount of metadata about an image before it can be properly georeferenced. This metadata, sometimes referred to as the *map parameters,* includes information such as the projection of the map. The projection conveys the manner in which the three dimensional features of the Earth have been distorted onto a two dimensional image. There are hundreds of map projections in use, and the way the distortion occurs influences the usefulness of the map in different domains. A list of nearly all map projections in use today can be found in [12]. Fortunately, organizations like the USGS and the Defense Mapping Agency use only a small subset of these.

Another map parameter used for georeferencing is the approximation of the Earth used in the projection. The Earth is not a perfect sphere, so it is first modeled as some ellipsoid, possibly a sphere, before being projected onto a plane. Other types of metadata are the *projection center* (where the distortion of the map is minimal) and the *insertion point* (a point on the image used to extrapolate data about other points on the image). The formulas used to do the georeferencing depend upon additional map parameters, increasing the amount of prerequisite knowledge.

I2K can perform all the basic georeferencing transformations provided the necessary metadata has been specified correctly. The georeferencing functionality of I2K is similar to other GIS software packages, and works with a range of underlying map types. The georeferencing feature in I2K was motivated by the need to enable statistical GIS data processing that is not available in other GIS software packages, for example, statistical measures of raster data (maps) over territories defined by vector data. The functionality of I2K has been tested thoroughly with standard examples from the literature [14], and the tests are described in the Section 5 of this document.

3. Georeferencing Transformations

For I2K, an interface has been developed to provide uniform georeferencing capabilities across a range of map types by abstracting away the specific parameters of different map types and maintaining a uniform and consistent internal representation. This design allows for the addition of as yet unsupported map types through the development of new code modules. In I2K, a *GeoImageObject* holds all georeferencing information. The fields with geographic referencing metadata in the GeoImageObject are populated when a new image is loaded, and dictate the algorithms applied during the transformation process.

A user of the I2K system is presented with a uniform interface regardless of the underlying representation. By encapsulating and rigorously testing the complex mathematical formulas, our system provides assurance that if the map parameters have been defined correctly, the georeferencing will also be correct.

The georeferencing system in I2K is built around coordinate transformations to and from three types of location representations: 2D map pixels (column and row), 3D lat/long coordinates, and 2D UTM values. These transformations are shown in Figure 3-1 and are implemented in a class called *GeoConvert*.

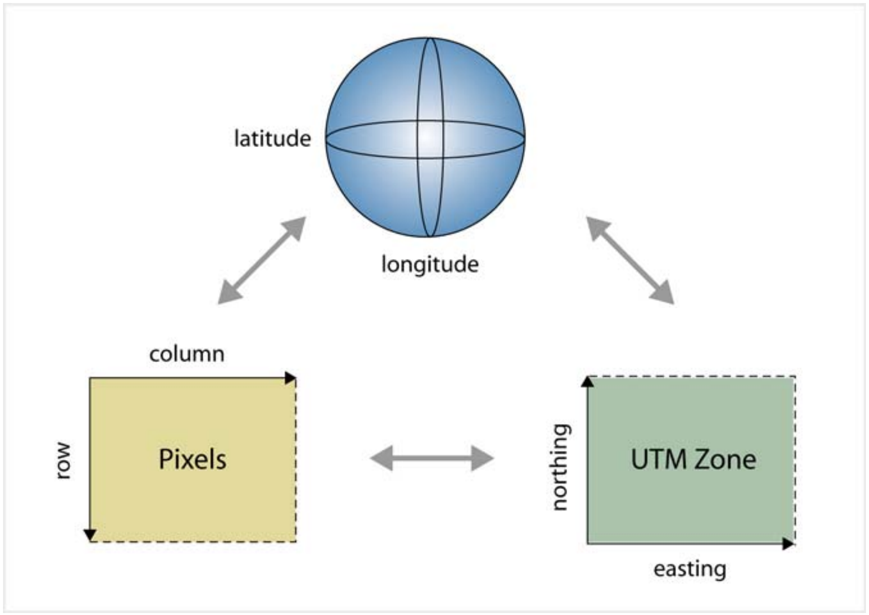


Figure 3-1: Coordinate transformations supported by I2K

The motivation for the transformations between 2D map and 3D lat/long coordinate systems is fairly obvious, because one would like to know the lat/long values of any pixel in a georeferenced image. However, the third location representation, UTM, is not familiar to most people.

The Universal Transverse Mercator (UTM) coordinate system is a Cartesian coordinate system developed by the oil industry and explained further in Appendix A. It is useful for specifying a number of points on a map without having to refer to latitude and longitude. Furthermore, UTM values facilitate metric distance calculations, which are difficult in lat/long coordinates where the distance between two adjacent degrees is dependent upon their location relative to the equator and the prime meridian. In UTM terminology, the horizontal (x) value is called *easting* and the vertical (y) coordinate is called *northing*. A UTM coordinate pair is called a *northing/easting* value.

3.1 I2K Model Types

As mentioned earlier, a *GeoImageObject* holds all georeferencing information in I2K and the fields of the object are populated with geographic referencing information when a new image is loaded. In order to perform the desired georeferencing transformations, an I2K *model type* must be set based on the map parameters that are loaded. The model type is not set if all the required metadata is not found and georeferencing cannot be preformed without the model type.

The term in georeferencing literature that is most analogous to the I2K model type is *datum*. A datum, when used in its strictest sense, refers to a complete specification of a mapping system including the ellipsoid, the map projection, and the coordinate system used [4]. This term seems to work well when referring to something like the OSGB 1936 datum that has an implied, official ellipsoid (Airy 1830). However, the term seems to have slipped into much weaker usage, sometimes only referring to a subset of the necessary attributes. Thus, we avoid the issue entirely and refer to *model type,* which encompasses all of the necessary attributes, throughout.

UTM coordinates are used in maps with various types of projections, but lend their name to a number of projections. Each projection type, along with the various parameters needed to perform the transformations, make up a model type in the I2K interface. Each model type has specific, private methods per model that can be called to perform the six transformations between pixel, UTM, and lat/long coordinates. The model types currently supported are:

1) The UTM Northern Hemisphere.
2) The UTM projection called the Ordnance Survey of Great Britain from 1936, OSGB 1936.
3) The Lambert Azimuthal Equal Area map projection.

The OSGB 1936 model type has been used primarily for testing purposes. Also, it should be noted that a sphere cannot accurately approximate the Earth. A sphere is only used as a model for the Earth on large-scale projections, such as the Lambert Azimuthal Equal Area projection. For any UTM projection, a standard ellipsoid should always be specified.

3.2 Transformations between Latitude / Longitude and UTM Northing / Easting

The UTM system is two-dimensional and the latitude/longitude system is three-dimensional, making transformations between them quite complex. The two methods in

GeoConvert that perform transformations between UTM and lat/long are based on equations in [5], [12], [13] and [14].

The transformation from UTM to lat/long is referred to as *Transformation 1,* and the transformation from lat/long to UTM is referred to as *Transformation 2*. The method descriptors for these transformations, together with brief explanations of the input and output parameters are given. The algorithms underlying these methods are chosen based on the metadata that was loaded into the GeoImageObject in conjunction with the image.

- **Transformation 1**

  public Point2DDouble **UTMNorthingEasting2LatLng**(Point2DDouble p)

  input: a Point2DDouble object: [northing, easting] - in meters

  output: a Point2DDouble object: [latitude, longitude] - in decimal degrees

- **Transformation 2**

  public Point2DDouble **LatLng2UTMNorthingEasting**(Point2DDouble p)

  input: a Point2DDouble object: [latitude, longitude] - in decimal degrees

  output: a Point2DDouble object: [northing, easting] - in meters

The model type determines whether the standard Molodensky equations ([5], [13], Appendix B) or the Lambert projection equations [14] are used to perform these two transformations. The Molodensky and Lambert equations require parameters for both the three-dimensional and two-dimensional coordinate systems. We present what must be known about the three-dimensional object and about the two-dimensional plane in the following subsections.

3.2.1 Ellipsoid Parameters to Define Latitude/Longitude Coordinates

Since the Earth is not a sphere, to obtain reasonable results an ellipsoid of some type is used as a model for the Earth. The geodetic ellipsoids used as models should always be referred to as *oblate ellipsoids of revolution*, but are referred to simply as *ellipsoids* in all georeferencing literature. An oblate ellipsoid of revolution is the ellipsoid formed by rotating an ellipse out of its plane around its minor (shorter) axis, as shown in Figure 3-2. All geodetic ellipsoids can be specified by two quantities: a semi-major axis and a semi-minor axis. If these two quantities are the same, the ellipsoid is a sphere.

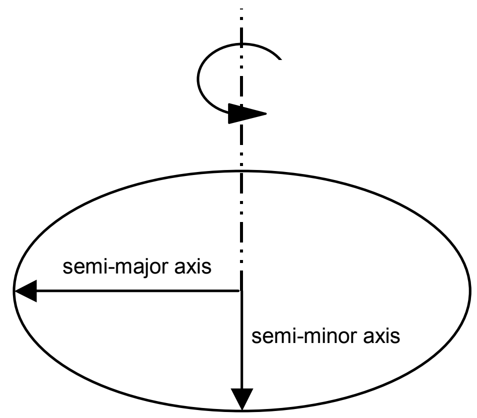


Figure 3-2: Ellipsoid parameters

Over approximately the past 200 years, many different measurements have been made of the Earth in an attempt to find the values for the semi-major and semi-minor axes that provide the most accurate representation of the Earth. Whenever a particular group made an official measurement of these values, they gave a name to the resulting ellipsoid. Dozens of these ellipsoids have been calculated and then discarded as newer ellipsoid models were developed. The names of these official ellipsoids are usually some combination of letters and numbers, such as WGS84 or GRS80.

The equations used in Transformations 1 and 2 make use of two values to define an ellipsoid: the semi-major axis and the square of the eccentricity of the ellipsoid. The *eccentricity* value, denoted as *e*, is a measure of the amount by which an ellipse varies from a circle. The square of the eccentricity is calculated with this equation:

$$e^2 = 2f - f^2$$

The *flattening* of the ellipse, *f,* is defined by:

$$f = (a - b)/a$$

In the flattening equation, *a* is the ellipsoid semi-major axis and *b* is the ellipsoid semi-minor axis. Often *(a - b)* is quite small compared to *a*, and it is common to provide the flattening quantity as *1/f*, or the *inverse flattening*. Because of these relationships, the required ellipsoid parameters can be derived from any one of three sets of information:

1) The radius of the sphere for the cases where a sphere is used for the model.
2) The semi-major axis and the semi-minor axis.
3) The semi-major axis and inverse flattening value.

3.2.2 Two-Dimensional UTM Map Parameters

In general, UTM map parameters should include information pertinent to a map's 2D coordinate system and information about the 3D to 2D projection used to obtain the map. Ideally, the map parameters will provide at least the following information: projection type, projection center, false easting, false northing, UTM zone, vertical resolution, horizontal resolution, and the tie point.

Of all map parameters, the map projection is perhaps the most important because it determines which set of equations is used to perform the transformations. Most of the other map parameters are used as inputs to that particular set of equations. The map projection is generally subsumed into the described in Section 3.1. Multiple model types might have the same map projection, but other map parameters could be different.

The map resulting from a projection will not have the same amount of distortion at each point. Different parts of the map plane are oriented to the Earth model differently, resulting in varying degrees of distortion. Generally, the distortion on a map will be minimal at its *projection center*. It is important to note that the projection center may or may not be in the viewable area of an image.

Two other quantities that are specific to a projection are the *false easting* and *false northing*. In a UTM map projection, points to the west or south of the projection center can have negative UTM values. In order to avoid the inconvenience introduced by negative values, false easting and false northing values are often added to computed northings and eastings to keep all values positive.

3.2.3 Fields for Lat/Long to UTM Transformation in I2K

In this section we enumerate the fields in the I2K GeoImageObject class that must be populated in order to perform any lat/long ↔ UTM transformations. For some specifications, the required information may be obtained from a number of different metadata combinations, corresponding to different I2K field combinations.

- **Projection Center of Map**

_GeoProjCenterLat:	The latitude value of the projection center.
_GeoProjCenterLng:	The longitude value of the projection center.
or	
_UTMZone:	A valid UTM zone specification, which will pinpoint the map projection center. See Appendix A for details.

- **False Northing/Easting**

_GeoProjFalseEasting:	The false easting value.
_GeoProjFalseNorthing:	The false northing value.

- **Ellipsoid Parameters**

_radius:	If a sphere is used as an approximation of the Earth, the radius of that sphere.

or

_majorAxis:	The semi-major axis value of the ellipsoid.
_minorAxis:	The semi-minor axis value of the ellipsoid.

or

_majorAxis:	The semi-major axis value of the ellipsoid.
_inverseFlat:	The reciprocal of *f*, the flattening of the ellipsoid.

or

_ellipsoidName:	A string that will automatically define the correct ellipsoid if set to one of several predefined values. Supported ellipsoids are: • WGS84 • WGS72 • GRS80 • Airy 1830

3.3 UTM Northing/Easting to and from Pixel Column and Row

It is also useful to go from UTM northing/easting to pixel-valued column and row. Since these are both horizontal/vertical 2-D coordinate systems, the transformations are linear equations mapping from one Cartesian system to another.

- **Transformation 3**

  public Point2DDouble **UTMNorthingEasting2ColumnRow**(Point2DDouble p)

  input: a Point2DDouble object: [northing, easting] - in meters

  output: a Point2DDouble object: [column, row] - pixel value

- **Transformation 4**

  public Point2DDouble **ColumnRow2UTMNorthingEasting**(Point2DDouble p)

  input: a Point2DDouble object: [column, row] - pixel value

  output: a Point2DDouble object: [northing, easting] - in meters

In the following subsections we discuss the I2K fields that must be populated for Transformations 3 and 4 to produce correct output.

3.3.1 Tie Point / Insertion Point

A *tie point,* also called an *insertion point*, is a pair of coordinates [(*i,j*), (*x,y*)] that ties together a pixel value (*i,j*) and its UTM equivalent (*x,y*). A tie point is actually specified as [(*i,j,k*), (*x,y,z*)] for use with three dimensional digital elevation maps, but the 3-D functionality is not yet supported in I2K. All two-dimensional images simply specify 0 for *k* and *z*.

The (*i,j*) coordinate is taken from pixel space with the *i* value specifying the column number of the pixel, and the *j* value specifying the row number of the pixel. The (*x,y*) value is taken from the UTM coordinate plane, with *x* specifying the easting value (horizontal measurement), and *y* specifying the northing value (vertical measurement).

To illustrate, a tie point might be [(0,0), (400000, 600000)], indicating that the pixel (0,0) has a UTM easting value of 400,000 meters and a UTM northing value of 600,000 meters. Note that the pixel space starts with (0,0) in the upper left corner of the image, with increasing column values to the right and increasing row values moving down, as indicated by the arrows in Figure 3-1.

3.3.2 Resolution / Scale

The other information required for Transformations 3 and 4 is the *resolution* or *scale* of the image. By knowing the number of meters represented by one pixel, a simple calculation can be performed to find the UTM values for any pixel, or the nearest pixel for any specified UTM value. In I2K documentation, the terms scale and resolution are sometimes used interchangeably, with a bias in favor of the term resolution. We prefer using the term resolution to avoid confusion with the *scaling factor* found in official ellipsoid descriptions.

To make the resolution as general as possible, horizontal and vertical values are usually specified separately. The horizontal resolution specifies the number of meters traversed when moving one column in pixel space. The vertical resolution specifies the number of meters traversed when moving one row in pixel space. Sample resolution values are 30 meters/pixel in the x direction (the horizontal resolution), and 40 meters/pixel in the y direction (the vertical resolution).

3.3.3 Parameters and Units of Measure

The requisite tie point and resolution parameters can be specified in meters for the tie point and meters/pixel for the resolution, or in terms of latitude/longitude degrees for the tie point and degrees/meter for the resolution. Examples 1 and 2 demonstrate these different units of measure:

- **Example 1**

Tie point:	[(0,0), (400000,600000)]	*UTM values are in meters.*
Resolution:	x: 30.0000 y: 40.0000	*Both values are in meters/pixel.*

- **Example 2**

Tie point:	[(0,0), (-87,45)]	*UTM values are in degrees.*
Resolution:	x: .00027778 y: .00027778	*Both values are in degrees/meter.*

Example 1, with meters and meters/pixel as the units of measure, is identical to the examples presented earlier in section 3.3.1 and 3.3.2. In Example 1, pixel (0,0) has a UTM easting value of 400,000 meters and a UTM northing value of 600,000 meters. One pixel in the x direction corresponds to 30 meters, while one pixel in the y direction corresponds to 40 meters.

Example 2 uses degrees and degrees/meter as the units of measure. In this example, the upper left corner of the image has lat/long value 87 degrees west longitude and 45 degrees north latitude. The resolution values have been nicely chosen because both x and y resolutions are one second of arc measurement. Recall that a second is 1/60th of a minute, which is 1/60th of a degree. Thus 1/3600th (.00027778) of a degree is a second in arc measurement. Taking the tie point and resolutions together, the pixel at column 1, row 1 has lat/long value 86 degrees, 59 minutes, 59 seconds west longitude and 44 degrees, 59 minutes, 59 seconds north latitude.

When presented with the units of measure shown in Example 2, a lat/long to UTM value conversion must take place prior to performing Transformations 3 and 4. I2K currently does this conversion automatically when it detects tie point/resolution information specified in degrees/minutes/seconds.

3.3.4 Fields for UTM to Pixel Transformation in I2K

In this section we enumerate the fields in the I2K GeoImageObject class that must be populated in order to perform Transformations 3 and 4. The required information may be obtained from a number of different metadata combinations, corresponding to different I2K field combinations.

- **Tie Point / Insertion Point**

  There are various sources for tie point information, which will be discussed further in Sections 4.1 and 4.2. Due to the naming inconsistency in the GIS domain and the potential for more than one source of georeferencing information, I2K populates multiple fields following several naming conventions when loading an image. These fields are later unified using a default priority to select among the field values when there are conflicting entries.

  The I2K fields related to the tie point for the various information sources are:

  Loaded from TIFF world file or an ERDAS IMG file

_rasterSpaceI, _rasterSpaceJ:	The *(i,j)* coordinates in pixel space.
_eastingInsertionValue:	The UTM easting value (*x*), in meters.
_northingInsertionValue:	The UTM northing value (*y*), in meters.

or

Loaded from TIFF private tag

_rasterSpaceI, _rasterSpaceJ:	The *(i,j)* coordinates in pixel space.
_rasterSpaceK:	The elevation of point *(i,j)* in pixel space.
_modelSpaceX:	The UTM easting value *(x)* in model space.
_modelSpaceY:	The UTM northing value *(y)* in model space.
_modelSpaceZ:	The elevation of *(x,y)* in model space.

- **Resolution / Scale**

  As was the case for the tie point, there are different sources for resolution information. The GeoImageObject fields related to these are shown, and they too are unified by I2K to resolve conflicts.

Loaded from TIFF world file or an ERDAS IMG file

_columnResolution:	The horizontal resolution, in meters.
_rowResolution:	The vertical resolution, in meters.

or

Loaded from TIFF private tag

_geoScaleX:	The horizontal resolution, in meters.
_geoScaleY:	The vertical resolution, in meters.

3.4 Latitude/Longitude to and from Pixel Column and Row

The most frequent application of georeferencing is to translate from a pixel value, that is, a column and row of an image, to a latitude/longitude pair. Transformation 5 performs this translation, while Transformation 6 does the reverse translation from a latitude/longitude pair to an image column and row.

- **Transformation 5**

  public Point2DDouble **ColumnRow2LatLng**(Point2DDouble p)

  input: a Point2DDouble object: [column, row] - pixel value output: a Point2DDouble object: [latitude, longitude] - in decimal degrees

- **Transformation 6**

  public Point2DDouble **LatLng2ColumnRow**(Point2DDouble p)

  input: a Point2DDouble object: [latitude, longitude] - in decimal degrees

  output: a Point2DDouble object: [column, row] – pixel value

These latitude/longitude ↔ column and row transformations are accomplished via calls to Transformations 1 through 4 discussed above. Specifically, Transformation 5 is carried out by a call to Transformation 4 followed by a call to Transformation 1. Calling Transformation 2 and then calling Transformation 3 accomplishes Transformation 6.

4. Sources of Georeferencing Information

In order to perform the geographic referencing transformations correctly, a number of map parameters must be given along with the map data. While some file formats were designed to contain georeferencing information, for example the ERDAS IMG file format, the file types of most digital maps were not designed to accommodate this metadata. For these file formats, of which the Tagged Image File Format (TIFF) is one example, the metadata must be provided in some manner for the map to be of any use. Although a number of approaches have been taken, the methods for including the metadata have not yet reached a point of standardization that enables a novice to quickly find the necessary information.

In this section we describe how geographic referencing information can be extracted from TIFF and IMG files, and detail the georeferencing information sources for the TIFF and IMG file formats supported by I2K.

4.1 TIFF Files

The georeferencing information extraction in I2K from TIFF world files (tfw) and/or private tags is based on the literature available in [10], [11]. The current solution for the TIFF format uses a combination of information extraction approaches that are based on three variants of TIFF files containing georeferencing information. Assuming that sufficient data is provided in some combination of the three sources, we can use the georeferencing interface described earlier. The three TIFF file variants are:

- One or more standardized files are distributed along with TIFF image data as .tfw and/or .txt files.
- The metadata is encoded in the image file using private TIFF tags.
- An extension of the TIFF format called *GeoTIFF* is used.

Given multiple information sources, it is possible to read conflicting values for the same geographic field. For example, tie points (Section 3.3.1) and resolution values (Section 3.3.2) can be specified in both the TIFF world file and the private TIFF tags. When both a tfw file and the private tags are present, the tfw values have priority.

Some georeferencing information may be specified either in deg/min/sec or UTM meters. In particular, if the information in private tag 33922 is specified in deg/min/sec, I2K will perform automatically the necessary transformation.

Finally, in the absence of necessary information, some transformations will not be executed. For example, if both the tfw file and the private tags 33550 and 33922 are missing, Transformations 3 and 4 cannot be performed.

4.1.1 TIFF World Files

Geospecific information for a map in the TIFF format is often included in separate file with the same root file name, but with the extension '.tfw' [10]. A tfw file is a short ASCII file that contains six values, four of those being the values discussed in Section 3.3.

Value	Description
30.000000	Column (horizontal) resolution
0.000000	Rotation in the horizontal direction
0.000000	Rotation in the vertical direction
-30.000000	Row (vertical) resolution
250000.00	Easting value of tie point
650000.00	Northing value of tie point

Table 4-1: Sample TIFF world file values

Table 4-1 shows typical values and descriptions for a TIFF world file. In looking at the tfw values, observe that the easting and northing values are available directly from the file. The horizontal and vertical resolution factors are also in the file, and in this example the vertical resolution is a negative value. A tfw file implies that the (*i*,*j*) value of the tie point is either the upper left corner or the lower left corner of the image. If the vertical resolution is given as a negative value, then the tie point of the image is implied to be the upper left, that is (0,0). If the vertical resolution is given as a positive value, then the tie point of the image is implied to be the lower left, that is (0, *number_of_rows*), where *number_of_rows* is the number of rows in the image.

The rotation factors of an image are useful if the image is somehow misaligned, but are not yet supported in I2K. It is also important to note that it is not possible to specify the additional *z* and *k* values of a tie point in a tfw file.

To load the georeferencing information from a tfw file into I2K, select a tfw file name from the I2K File menu. When a tfw file name is selected, I2K will load both the tfw and tif files that represent a digital map.

4.1.2 Private TIFF Tags

The TIFF file format uses a data structure known as a *tag* to supply information about an image. Every tag within the TIFF specification has a particular number associated with it, called its *value*. Any software that reads a TIFF file reads a tag structure by first examining the tag value, and then interpreting the meaning of the tag based on the tag value.

The TIFF specification [7] defines the values and meanings of every valid tag, so no single entity can create a TIFF tag of arbitrary value for its exclusive use. However, it is possible for a company to register one or more TIFF tags for private use, and a number of these so-called private tags are quite useful for geographic referencing [11]. Four private tags are supported in I2K.

- **Private Tag 33550**

Private tag 33550 is referred to as the *ModelPixelScaleTag* and consists of three IEEE double precision floating-point numbers as shown in the following table. This structure has built-in support for digital elevation maps, but non-zero *z* resolution is not yet supported in I2K.

Private Tag 33550		
Standard Name	**Description**	**I2K Field**
ScaleX	Horizontal (*x*) resolution	_geoScaleX
ScaleY	Vertical (*y*) resolution	_geoScaleY
ScaleZ	Elevation (*z*) resolution	_geoScaleZ

- **Private Tag 33922**

Private tag 33922 is referred to as the *ModelTiePointTag*. It usually consists of six IEEE double precision floating-point numbers organized as shown in the following table. Tag 33922 may contain a sequence of [*(I,J,K), (X,Y,Z)*] values, specifying multiple tie points, but only the first is used for the transformations in I2K.

Private Tag 33922		
Standard Name	**Description**	**I2K Field**
I	Column number of tie point in pixel space	_rasterSpaceI
J	Row number of tie point in pixel space	_rasterSpaceJ
K	Elevation value of pixel (*I,J*)	_rasterSpaceK
X	Easting value in model space	_modelSpaceX
Y	Northing value in model space	_modelSpaceY
Z	Elevation of (*X,Y*) in model space	_modelSpaceZ

- **Private Tag 34735**

Private tag 34735 is called the *GeoKeyDirectoryTag*. This tag is used to implement the GeoKey data structure for GeoTIFF files, an extension of TIFF. See Section 4.1.3 for details.

- **Private Tag 34737**

Private tag 34737 is referred to as the *GeoAsciiParamsTag*. This tag holds the ASCII values of any GeoKeys that contain ASCII characters. An example is GeoKey 3073. See Section 4.1.3 for details.

4.1.3 GeoTIFF Keys

The private TIFF tag solution worked well for the individual companies who registered the tags. However, for third-party GIS companies, the solution was not ideal because their software had to interoperate with each company's standards. The third-party companies wanted a public standard that would be platform independent and available to anyone. The GeoTIFF format is an open standard built from the TIFF 6.0 specification. Its purpose is to include geographic information in a TIFF document in the form of GeoTIFF keys.

GeoTIFF keys, commonly known as *GeoKeys*, can be described as data structures built on top of the tag structure. A GIS reader that does not support the GeoKey specification reads all GeoKeys simply as TIFF tags it does not understand. Following the example of the TIFF tag, each GeoKey has a unique value called an *ID number*. By efficiently using the TIFF tag structure, the GeoTIFF standard allows for GeoKeys with ID numbers ranging from 0 to 65,535 while using a maximum of six registered tags. Two of these registered tags were listed in Section 4.1.2.

- **GeoKey 1024**

GeoKey 1024 is called the *GTModelTypeGeoKey*. It consists of one short (2-byte unsigned integer) value that specifies the general type of coordinate system used. Geographic and geocentric coordinate system images are less common than images with a projection coordinate system. Images with geographic and geocentric coordinate systems are not supported in I2K.

GeoKey 1024	
Possible Value	**Description**
1	A projection coordinate system is being used
2	A geographic coordinate system is being used
3	A geocentric coordinate system is being used

- **GeoKey 1026**

GeoKey 1026 is the *GTRasterTypeGeoKey*. The purpose of this key is to remove the ambiguity inherent in representing a coordinate system with raster data. The key contains one short (2-byte unsigned integer) with two possible values that indicate how to interpret a raster coordinate (a pixel).

GeoKey 1024	
Possible Value	**Description**
1	Raster pixel represents area
2	Raster pixel represents point

As an example, consider a raster coordinate of (1,1). If the value of GeoKey 1026 is 1, it indicates that the raster coordinate (1,1) refers to the entire square grid with boundaries (0,0) in the upper left and (1,1) in the lower right. More generally, in this system a coordinate (*i,j*) refers to the 1 unit square area with (*i,j*) as its lower right corner. In contrast, if the value of GeoKey 1026 is 2, a pixel value refers only to point indicated by the intersection in pixel space of a vertical line *i* units to the right of the origin with a horizontal line *j* units down from the origin. In our example for raster coordinate (1,1), this would be the point 1 unit right from the origin and 1 unit down. For details see Section 2.5.2.2 in [8].

- **GeoKey 3072**

GeoKey 3072 is the *ProjectedCSTypeGeoKey*. If GeoKey 1024 has value 1, then this key indicates the specific projection that was used. There are 12,761 allowable values for this key, ranging from 20000 to 32760, with each one a well-defined projection on a particular geographic location. See Section 6.3.3.1 in [8] for details.

- **GeoKey 3073**

The *PCSCitationGeoKey*, GeoKey 3073, exists to compensate for the fact that GeoKey 3072 is not a user-friendly way to specify a map projection. The value of GeoKey 3073 is an ASCII string intended to give the user understandable information about the map projection. Often it consists of just a few words providing information such as the UTM zone (see Appendix A), the ellipsoid used, or other geographic metadata about the image. The string is shown to the user with other geographic information in the *GeoInfo* window in I2K (see Section 5).

4.2 ERDAS IMG Files

I2K is also capable of georeferencing maps stored in the ERDAS IMG format. For these files, the metadata is always included in a separate header file. The header file has the same root file name as the image, but with '.hdr' as its extension. Many types of metadata about the img file are included in the header file in a ‘keyword = value’ format, with the important keyword for geographic referencing being *map info*.

The map info entry in the header file has the format:

map info = { *projection_name, i, j, x, y, horizontal_resolution, vertical_resolution UTM_zone, ‘North’* or *‘South’*}

The individual values in the map info entry are:

projection_name:	The type of projection used in the image, often UTM. See Appendix A.
[(*i,j*),(*x,y*)]:	The tie point. See Section 3.3.1.
horizontal_resolution:	The horizontal resolution of a pixel. See Section 3.3.2.
vertical_resolution:	The vertical resolution of a pixel. See Section 3.3.2.

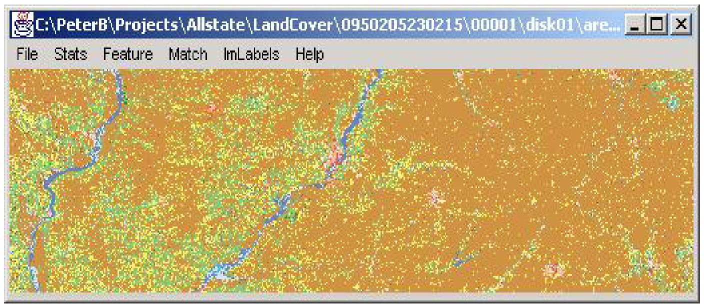
The last two individual values are only present if the projection specified is the UTM Northern Hemisphere or the UTM Southern Hemisphere projection:

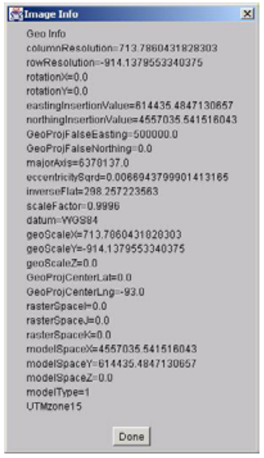
UTM_zone:	The UTM zone in which the region is located. See Appendix A.
‘North’ or *‘South’:*	Indicates if the image is in the Northern or Southern Hemisphere of the specified zone. See Appendix A.

5. Georeferencing in I2K

The I2K software enables two primary functions related to georeferencing files. First, it allows a user to view georeferencing information for any loaded map. Second, it uses georeferencing to register raster files, such as maps, and vector files, such as boundaries or contours, stored in the shape file format.

Once a georeferenced image is loaded in I2K, a user can view all of its available geospecific information by right clicking on the image and selecting *GeoInfo* from the menu. The values displayed will normally include the scale/resolution of the image, the projection type of the map, and/or the ellipsoid name from which this map was projected.


C:\PeterB\Projects\Allstate\LandCover\0950205230215\00001\disk01\are...
File Stats Feature Match ImLabels Help



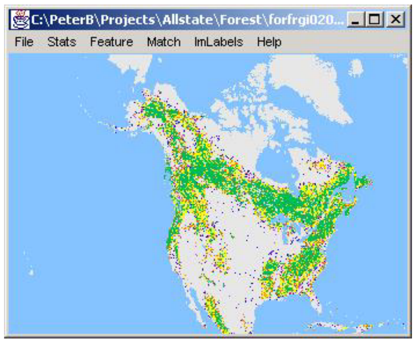
```
Geo Info
columnResolution=713.7860431828303
rowResolution=-914.1379553340375
rotationX=0.0
rotationY=0.0
eastingInsertionValue=614435.4847130657
northingInsertionValue=4557035.541516043
GeoProjFalseEasting=500000.0
GeoProjFalseNorthing=0.0
majorAxis=6378137.0
eccentricitySqrd=0.0066943799901413165
inverseFlat=298.257223563
scaleFactor=0.9996
datum=WGS84
geoScaleX=713.7860431828303
geoScaleY=-914.1379553340375
geoScaleZ=0.0
GeoProjCenterLat=0.0
GeoProjCenterLng=-93.0
rasterSpaceI=0.0
rasterSpaceJ=0.0
rasterSpaceK=0.0
modelSpaceX=4557035.541516043
modelSpaceY=614435.4847130657
modelSpaceZ=0.0
modelType=1
UTMzone15
```

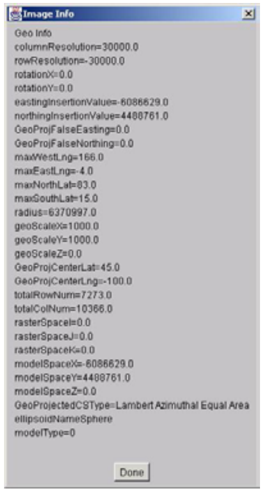
Figure 5-1: A land cover map image and its georeferencing information

An example is illustrated in Figure 5-1.

A specific I2K tool that uses georeferencing functionality is the *GeoFeature* option on the Tools menu. With this option, a user can choose to register and overlay a contour or boundary stored in a shape file onto the georeferenced image. If enough image metadata are correctly loaded to support all six transformations, the shape file with points specified by latitude and longitude will be correctly georeferenced over the image. Once the geo-registration has been completed, statistics of image attributes over each boundary can be computed. An example of geo-registration is shown in Figure 5-2.

All georeferencing functionality has been tested and can be rechecked at any time with the *Test_GeoConvert* class in the *test.libGeo* package. The Test_GeoConvert class creates a GeoImageObject and tests all six transformations for each model type. If any source in the GeoConvert class is changed, running the main method in Test_GeoConvert will verify that the transformations are still working correctly. The output for each transformation test is compared to reference values. If the difference between the test and reference values is negligible, the model passes the test for that transformation.





```
Geo Info
columnResolution=30000.0
rowResolution=-30000.0
rotationX=0.0
rotationY=0.0
eastingInsertionValue=-6086629.0
northingInsertionValue=4488761.0
GeoProjFalseEasting=0.0
GeoProjFalseNorthing=0.0
maxWestLng=166.0
maxEastLng=-4.0
maxNorthLat=83.0
maxSouthLat=15.0
radius=6370997.0
geoScaleX=1000.0
geoScaleY=1000.0
geoScaleZ=0.0
GeoProjCenterLat=45.0
GeoProjCenterLng=-100.0
totalRowNum=7273.0
totalColNum=10366.0
rasterSpaceI=0.0
rasterSpaceJ=0.0
rasterSpaceK=0.0
modelSpaceX=-6086629.0
modelSpaceY=4488761.0
modelSpaceZ=0.0
GeoProjectedCSType=Lambert Azimuthal Equal Area
ellipsoidNameSphere
modelType=0
```

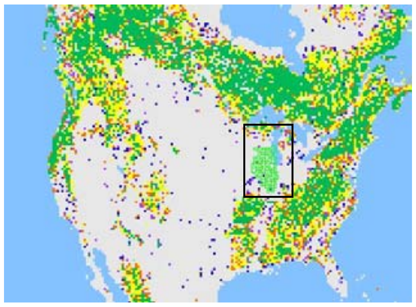


Figure 5-2: A TIFF image with forest cover labels (left) and its georeferencing information (middle) can be geo-registered with a shape file containing boundaries of Illinois counties (right image – Illinois falls within indicated rectangle).

It should also be noted that loading a sub-area or a sub-sampled image requires adjustment of georeferencing parameters. These two loading scenarios are accommodated in I2K via the *OpenArea* and *OpenLarge* options accessible from the File menu. The georeferencing parameters are adjusted so that the resolution and spatial bounding box match the loaded image data. Maps can be saved via the *SaveAs* option accessible from the File menu if the user specifies a name with the .tfw suffix.

6. Summary

I2K provides georeferencing for several of the most common types of digital map data. The implementation allows new projection and file types to be supported with minimal additional coding effort. Considerable effort was spent on effectively reading and utilizing the available map metadata.

To keep the software as easy to use as possible, I2K attempts to find the required data in several places before giving up, and occasionally makes a 'best guess' if the necessary data cannot be found. The result of our work is the ability to load georeferenced images with the geospecific information embedded in the image file, included in standard additional files, or some combination of these two. In spite of our conscious effort to design a flexible and extensible system, there are undoubtedly formats that I2K will not be able to properly georeference. Preparing for every possible form of metadata that I2K could encounter would be an exhaustive task.

A possible future direction for geographic referencing in I2K would be the addition of an Expert Mode. If the user knows required information about a map image whose metadata is in a different format than I2K expects, the user could simply use a GUI tool to tell I2K the necessary information. This GUI tool could take a 'wizard' format, prompting the user with a series of questions to facilitate the data entry. The direct-entry feature would require more user knowledge and expertise, but would provide immediate accessibility to the georeferencing capabilities of I2K for images in formats not yet coded into the system.

I2K could also be extended to add support for more map projections. The formulas to perform the required six transformations for many more projections are found in various texts [9], [12] and have been reprinted in technical documents found online [15].

References

[1] Gateway site to entire topic: http://www.remotesensing.org
[2] ESRI website: http://www.esri.com/
[3] USGS Mapping Science Software:
http://mapping.usgs.gov/www/products/software.html
[4] Educational site: http://www.colorado.edu/geography/gcraft/notes/datum/datum.html
[5] Technical Information:
http://www.posc.org/Epicentre.2_2/DataModel/ExamplesofUsage/eu_cs.html
[6] Research Systems, Inc, The Environment of Visualizing Images (ENVI),
http://www.rsinc.com/envi/index.cfm
[7] TIFF Revision 6.0. Aldus Corporation. Copyright 1986-88, 1992.
[8] GeoTIFF Revision 1.0. Ritter, Niles. Ruth, Michael. 2000.
[9] *Map Projections – A Reference Manual.* L. M. Bugayevsky and J. P. Snyder. Taylor and Francis, 1995.
[10] Information on tfw files:
http://www.genaware.com/html/support/faqs/imagis/imagis15.htm
[11] Information on private TIFF tags:
http://remotesensing.org/geotiff/spec/geotiff2.6.html
[12] *Understanding Map Projections*. Kennedy, Melita. Kopp, Steve. Environmental Systems Research Institute, Inc.
[13] Molodensky equations reference
http://www.posc.org/Epicentre.2_2/DataModel/ExamplesofUsage/eu_cs34h.html
[14] Lambert projection equations
http://mathworld.wolfram.com/LambertAzimuthalEqual-AreaProjection.html
[15] Online reference to transformation equations
http://www.posc.org/Epicentre.2_2/DataModel/ExamplesofUsage/eu_cs.html

Appendix A. UTM Northern Hemisphere Project

The purpose of this appendix is to give a synopsis of a projection that is commonly used for current maps of regions within the continental United States.

A.1 Transverse Mercator Projection

In the Transverse Mercator Projection, a cylinder passes through the Earth model, slicing through the ellipsoid and then emerging on the other side. Any line of longitude can be chosen to be in the center of the piece of the ellipsoid that sits on the outside of the cylinder. The ellipsoid surface is then projected onto the cylinder and the unfolded cylinder then becomes our map.

The Transverse Mercator Projection is shown graphically in Figure A-1. Notice in figure that the ends of the cylinder are quite far from the ellipsoid surface, causing a significant amount of distortion. The most accurate (and useful) part of our map is the exposed piece in the center. Its distance from the cylinder is small so the distortion is minimized at these points. This ‘slicing’ of our ellipsoid leads naturally to the concept of UTM zones, discussed in the next section of Appendix A.

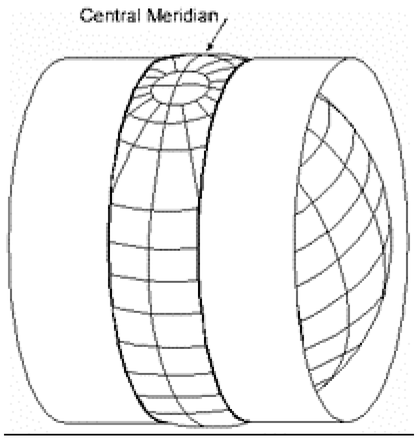


Figure A-1: Transverse Mercator Projection

A.2 UTM Zones

The Transverse Mercator projection can provide a highly accurate map for only a relatively small span of longitude at any one time. To accurately cover the entire planet with this projection, one could make a projection for a certain ‘slice’ of the ellipsoid, rotate the ellipsoid, make another projection, rotate again, and so on. The only piece of each projection used would be the center slice, but after making many slices, one would get the entire Earth. To that end, the UTM Zone system was developed to standardize this slicing.

In the UTM Zone system, the Earth is split up into 60 zones, each 6 degrees of longitude wide. The center longitude of each zone serves as the projection center for a map of any region inside that zone. Any map using the UTM Northern Hemisphere projection would then only show a region within one particular zone. As demonstrated by Figure A-2, it is often impossible to show an entire state and remain completely within a single UTM zone. When this occurs, the tie point defined in Section 3.3.1 determines which UTM zone will be used for the given image.

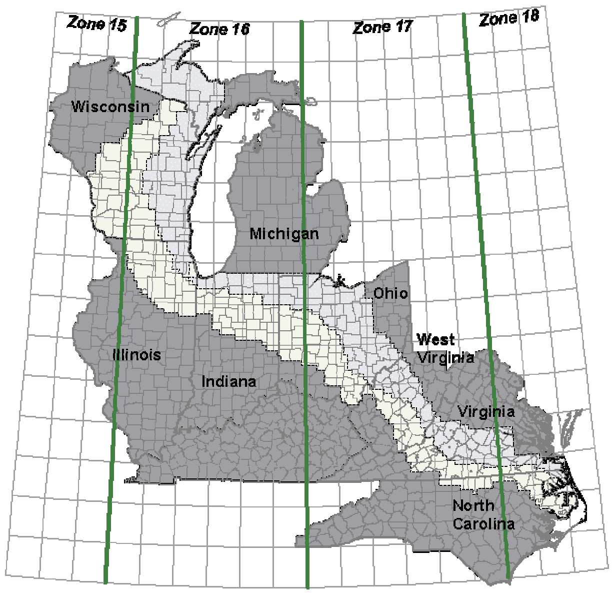


Figure A-2: The UTM zones for central United States

Appendix B. Molodensky Equations

In this appendix we present both the *forward* and *reverse* Molodensky equations used for geographical transformations 1 and 2. We make heavy use of [13] in this appendix, adding additional comments for clarity. The forward Molodensky equations take a latitude/longitude value to its northing/easting pair, and the reverse equations take a northing/easting pair to its latitude/longitude value.

B.1 Forward Equations

The forward Molodensky equations can be thought of as a function, *F*, which takes a latitude and longitude, along with all of the parameters of the UTM projection, and returns the northing and easting. *F* is specified as:

$$F\colon\ (\phi, \lambda, \phi_0, \lambda_0, k_0, a, e^2, FE, FN) \rightarrow\ (N, E)$$

The parameters for *F* are:

ϕ:	the latitude
λ:	the longitude
ϕ_0:	the projection center latitude
λ_0:	the projection center longitude
k_0:	the scale factor of the projection
a:	the semi-major axis of the ellipsoid
e^2:	the square of the eccentricity of the ellipsoid
FE:	the false easting of the projection
FN:	the false northing of the projection
N:	the northing
E:	the easting

The function *F* makes use of two equations that compute easting and northing values:

$$E = FE + k_0\,\nu \cdot \left[A + (1 - T + C)\frac{A^3}{6} + (5 - 18T + T^2 + 72C - 58e'^2)\frac{A^5}{120}\right]$$

$$N = FN + k_0 \cdot \left\{M - M_0 + \nu \tan\phi \cdot \left[\begin{array}{l}\dfrac{A^2}{2} + (5 - T + 9C + 4C^2)\dfrac{A^4}{24} \\ + (61 - 58T + T^2 + 600C - 330e'^2)\dfrac{A^6}{720}\end{array}\right]\right\}$$

The equation parameters *T, C,* and *A* are specified as follows:

$$T = \tan^2 \phi$$

$$C = \frac{e^2}{1-e^2}\cos^2 \phi = e'^2 \cos^2 \phi$$

$$A = \left(\lambda - \lambda_0\right)\cos\phi$$

The parameter e'^2 is defined as:

$$e'^2 = e^2 / \left[1 - e^2\right]$$

M is defined here, with M_0 calculated in the same manner substituting ϕ_0 for ϕ:

$$M = a \cdot \left[\begin{array}{l} \left(1 - \frac{e^2}{4} - \frac{3e^4}{64} - \frac{5e^6}{256} - \ldots\right)\phi \\ - \left(\frac{3e^2}{8} + \frac{3e^4}{32} + \frac{45e^6}{1024} + \ldots\right)\sin 2\phi \\ + \left(\frac{15e^4}{256} + \frac{45e^6}{1024} + \ldots\right)\sin 4\phi \\ - \left(\frac{35e^6}{3072} + \ldots\right)\sin 6\phi + \ldots \end{array}\right]$$

The *v* term in the easting formula is defined by the v_1 formula presented in the following section on the Reverse Equations, using ϕ instead of ϕ_1.

B.2 Reverse Equations

The reverse Molodensky equations require the same projection parameters as the forward equations, but take a northing/easting and return a latitude and longitude. They can be thought of as a function *R* specified as:

$$R\text{:}\ (N, E, \phi_0, \lambda_0, k_0, a, e^2, FE, FN) \rightarrow (\phi, \lambda)$$

The parameters in this *R* reverse function are the same as those described earlier for the *F* forward function.

The function *R* makes use of two equations that compute the latitude and longitude:

$$\phi = \phi_1 - \frac{\nu_1 \tan \phi_1}{\rho_1}\left[\begin{array}{l} \frac{D^2}{2} - \left(5 + 3T_1 + 10C_1 - 4C_1^2 - 9e'^2\right)\frac{D^4}{24} \\ + \left(61 + 90T_1 + 298C_1 + 45T_1^2 - 252e'^2 - 3C_1^2\right)\frac{D^6}{720} \end{array}\right]$$

$$\lambda = \lambda_0 + \left[\begin{array}{l} D - \left(1 + 2T_1 + C_1\right)\frac{D^3}{6} \\ + \left(5 - 2C_1 + 28T_1 - 3C_1^2 + 8e'^2 + 24T_1^2\right)\frac{D^5}{120} \end{array}\right] \Big/ \cos \phi_1$$

The equation parameters v_1 and p_1 are defined as:

$$\nu_1 = \frac{a}{\sqrt{1 - e^2 \sin^2 \phi_1}}$$

$$\rho_1 = \frac{a\left(1 - e^2\right)}{\left(1 - e^2 \sin^2 \phi_1\right)^{3/2}}$$

The equation parameters *D*, *T*₁, and *C*₁ are defined as:

$$D = \frac{E - FE}{\nu_1 k_0}$$

$$T_1 = \tan^2 \phi_1$$

$$C_1 = e'^2 \cos^2 \phi$$

The parameter ϕ_1 is defined by the sum:

$$\begin{aligned}\phi_1 = \mu_1 &+ \left(3e_1/2 - 27e_1^3/32 + \ldots\right)\sin 2\mu_1 \\ &+ \left(21e_1^2/16 - 55e_1^4/32 + \ldots\right)\sin 4\mu_1 \\ &+ \left(151e_1^3/96 + \ldots\right)\sin 6\mu_1 \\ &+ \left(1097e_1^4/512 - \ldots\right)\sin 8\mu_1 + \ldots\end{aligned}$$

The parameters e_1 and μ_1 are defined as:

$$e_1 = \frac{1 - \left(1 - e^2\right)^{1/2}}{1 + \left(1 - e^2\right)^{1/2}}$$

$$\mu_1 = \frac{M_1}{a \cdot \left(1 - e^2/4 - 3e^4/64 - 5e^6/256 - \ldots\right)}$$

Finally, M_1 is defined in terms of M_0 from above:

$$M_1 = M_0 + \left(N - FN\right) / k_0$$