Mapping the EXIF Data Stored in an Image

Greg Pagett
University of Nebraska at Omaha
casperpage@gmail.com

This is the second of three Practical Cartographer's Corner pieces in this issue which come from a graduate course entitled "Cartographic Methods," taught by Michael Peterson (mpeterson@unomaha.edu) at the University of Nebraska at Omaha during the 2013 Spring semester.

Almost all pictures taken with mobile phones incorporate location within a portion of the image file. These Exchangeable Image File Format (EXIF) data are a standard part of each picture, and include a variety of information, including the device used, the lens, the exposure, and the date and time the image was taken (Wikipedia 2013). If the phone can determine its location through GPS, WiFi or cell phone tower triangulation, the latitude and longitude will also be captured.

The process of extracting location data is quite simple, and my purpose here is to give an overview of the necessary code and some of the options for displaying them. There are few requirements: the image must have an Internet address and must include the location within the EXIF part of the file. One thing to note is that some sites, such as Flickr and Facebook (Bailey 2010), often extract the EXIF data and then remove that information from the image; you will need the original images and not ones uploaded to Flickr or other similar sites.

PHP was used for extracting location data, as it has many built-in functions that are useful for this task. The PHP code in Example 1, checks to make sure there is locational data stored in the EXIF. Next, it extracts the data and does the conversion necessary to get the GPS coordinates into the proper latitude and longitude format. The data are stored as degrees, minutes, and seconds and need to be converted into degree decimal format. Finally, the function returns the properly formatted latitude and longitude. This point can then be placed on a map.

The PHP code in Example 2 places the latitude and longitude into the HTML code:

Line 1 — calls the function `readGPSinfoEXIF()` and sets the variable `$results` to the values returned.

Line 2 — sets the `$lat` variable equal to the value in the first element of the `$results` array.

Line 3 — sets the `$lng` variable to the second element in the `$results` array. It is then multiplied by -1 to put the point in the proper hemisphere.

Line 4 — PHP outputs the HTML values for the body tag and inserts the call to the JavaScript function `initialize()` with the values for latitude and longitude returned from PHP variables.

Line 5 — PHP outputs the HTML code to create a table section and creates the first table row.

Line 6 — PHP outputs the code to create the first standard cell in the table and inserts the map variables.

Line 7 — PHP outputs the standard table cell that will contain the image that contains the locational data.

Line 8 — Closes the table row and table.

In Example 3, the JavaScript function `initialize()` creates the actual map and assigns the values to the title and info window. By querying the Google geocoder function with the latitude and longitude, the `initialize()` function can determine the closest address. The results are returned in a number of formats. For this example, the basic address, city, state, and ZIP information are returned. This reverse geocoding is only an estimate and is not guaranteed to be the closest address (Google 2013).

```
function readGPSinfoEXIF()
{
      $exif=exif_read_data('outside.jpg', 0, true); //sets a variable with all
                                                    //the EXIF data

     if(!$exif || $exif['GPS']['GPSLatitude'] == '') //Determines if the
                              //geolocation data exists in the EXIF data
     {
          return false;             //no GPS Data found
          echo "No GPS DATA in EXIF METADATA";
       }
     else
     {
            $lat_ref = $exif['GPS']['GPSLatitudeRef'];
            $lat = $exif['GPS']['GPSLatitude'];        //sets a variable equal
                                                      //to the Latitude
            list($num, $dec) = explode('/', $lat[0]);  //calculates the Degrees
            $lat_s = $num / $dec;
            list($num, $dec) = explode('/', $lat[1]);  //calculates the Minutes
            $lat_m = $num / $dec;
            list($num, $dec) = explode('/', $lat[2]);  //calculates the Seconds
            $lat_v = $num / $dec;

            $lon_ref = $exif['GPS']['GPSLongitudeRef'];
            $lon = $exif['GPS']['GPSLongitude'];        //sets the variable for
                                                       //the longitude
            list($num, $dec) = explode('/', $lon[0]);   //puts the degrees into
                                                       //a variable
            $lon_s = $num / $dec;
            list($num, $dec) = explode('/', $lon[1]);   //puts the minutes into
                                                       //a variable
            $lon_m = $num / $dec;
            list($num, $dec) = explode('/', $lon[2]);   //puts the seconds into
                                                       //a variable
            $lon_v = $num / $dec;

            //Calculates the GPS location in decimal form.
              $gps_int = array($lat_s + $lat_m / 60.0 + $lat_v / 3600.0, $lon_s
                     + $lon_m / 60.0 + $lon_v / 3600.0);
            return $gps_int;   //returns the coordinates
       }
}
```

Example 1: PHP code for extracting the EXIF data from an image.

Example 2: This code writes the HTML that displays the web page with the values gathered from the EXIF extract. The line numbers are only used for the line-by-line explanation.

```
Line 1  - $results = readGPSinfoEXIF();
Line 2  - $lat = $results[0];
Line 3  - $lng = $results[1] * -1; //the returned value for longitude
            //must be negative to be mapped on this side of the world.
Line 4  - echo "<body onload=\"initialize( $lat , $lng )\">" ;
Line 5  - echo "<table><tr>";
Line 6  - echo " <td><div id=\"map-canvas\" style=\"width: 800px; height:
                  480px;\"></div> </td>";
Line 7  - echo " <td><image src='outside.jpg' width=200: height=300><td>";
Line 8  - echo " </tr></table>";
```

```
<script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false">
  </script>
<script>
  var geocoder;
  var map;
  function initialize(lat, lng) {                          //name of the function
       geocoder = new google.maps.Geocoder();              //creates a new geocoder object
       var latlng = new google.maps.LatLng(lat, lng); //creates new lat and long object
       var mapOptions = {
       zoom: 11,                                                  //sets the zoom level to 11
       center: latlng,                                            //makes the center of the map
                                        //the latitude and longitude of the image
       mapTypeId: google.maps.MapTypeId.ROADMAP   //sets the type of map to be
                                                  //displayed
       }
              //creates the actual map object with the previously set options
       map = new google.maps.Map(document.getElementById("map-canvas"), mapOptions);

       geocoder.geocode({'latLng': latlng}, function(results, status)
       {
       if (status == google.maps.GeocoderStatus.OK) {     //logic chekt to see if
                                                           // mapping successful
                     if (results[1]) {
                     marker = new google.maps.Marker({          //create marker at the
                                                            //point from the image
                            position: latlng,
                                   title: results[0].formatted_address, //create the title
                            //of the map the same as the closest address found.
                            map: map});
                     var infowindow = new google.maps.InfoWindow({ //create an infowindow
                                                          //to display the address also
                            content: results[0].formatted_address});
                     infowindow.open(map,marker);
              }
              } else {
              alert("Geocoder failed due to: " + status);
       }
        });
}
</script>
```

Example 3: The JavaScript used to get the geocoded information based on the latitude and longitude returned from the PHP code.

Now that the locational data have been extracted and the map has been created, the information is sent to the client and the point is mapped. The information is displayed in both the information window and the title, which displays when the mouse hovers over the marker.

Adjusting the display marker is possible. For example, a thumbnail image could be used to substitute the image for the marker. The image used as the marker needs to be small and should be in the Portable Network Graphics (PNG) format.

```
var image = 'beachflag.png';
var myLatLng = new google.maps.LatLng(-33.890542, 151.274856);
var beachMarker = new google.maps.Marker({
    position: myLatLng,
    map: map,
    icon: image
});
```

Example 4: The JavaScript code to create a custom marker icon.

In Example 4, `icon: image` is the code that assigns the image to the marker. At this point, we can also add a shadow with the code `shadow: shadowimage` where shadowimage is the location and name of the shadow file. This shadow file is usually a few pixels larger than the thumbnail image. One could also set the shadowimage to a grey image if the location is found

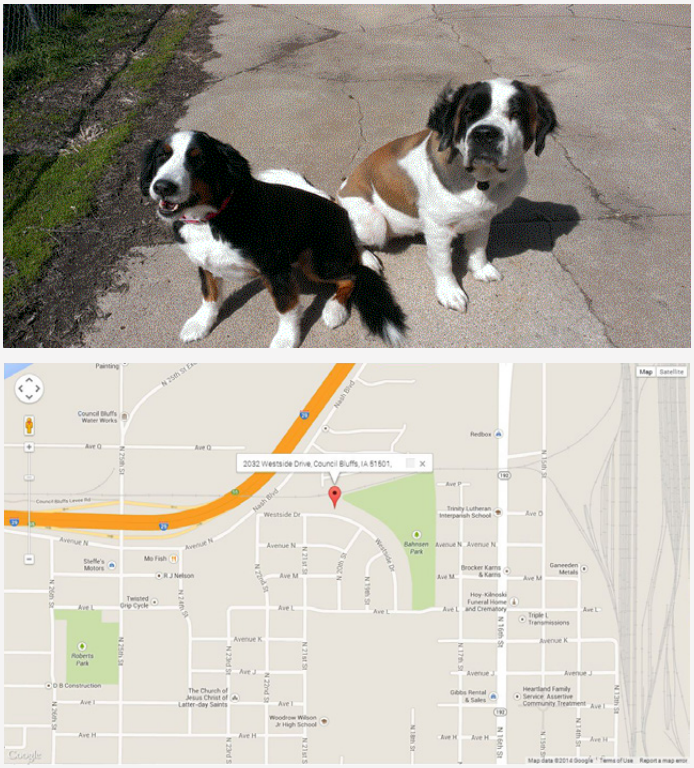

Figure 1: The image and its location based on data extracted from the EXIF part of the image file.

in the EXIF data, or alternatively the shadowimage could be red to indicate missing data (Figure 2).

Other forms of symbolization are possible. For instance, if the ZIP code is missing from the Google query, one could change the image to bounce with the `animation: google:maps.Animation.BOUNCE` code in the marker settings.

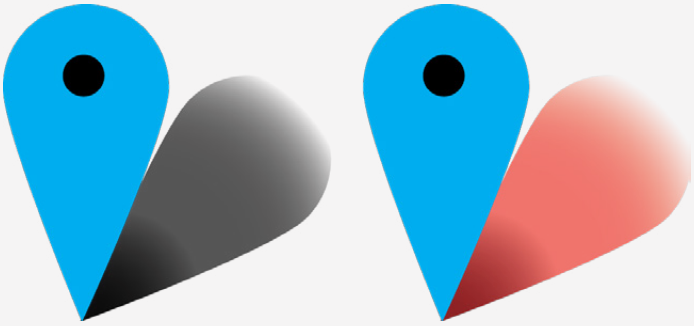

Figure 2: The first image is an example of a shadow that would represent the existence of EXIF data for location. The second image, using a red shadow, represents missing data.

There are many options available for displaying EXIF data that would allow the developer to add uniqueness and interactivity to their maps through images. The EXIF data are currently used in many popular applications and websites. The examples here use Google's API but this is not the only mapping API currently available. Leaflet and OpenStreetMap can both display the images and their respective EXIF data in a similar manner to Google.

In the end, it is up to the developer to decide which options to use and how to utilize the available API tools, and as geocoding becomes more popular, more tools and mapping options will become available

With some imagination, it is easy to create a unique mapping experience with photos and EXIF data.

REFERENCES

Bailey, Jonathan. 2010. "Flickr and Facebook STILL Strip EXIF Data." Plagarism Today. Accessed April 14, 2013. http://www.plagiarismtoday.com/2010/04/22/flickr-and-facebook-still-strip-exif-data/.

Google. 2013. "Reverse Geocoding (Address Lookup)." *Google Developers*. Accessed April 14. https://developers.google.com/maps/documentation/geocoding/#ReverseGeocoding.

Wikipedia. 2013. "Exchangeable image file format." Accessed April 14. http://en.wikipedia.org/wiki/Exchangeable_image_file_format.