Paper RIV-09

Geocoding Crashes in Limbo

Carol Martell and Daniel Levitt
Highway Safety Research Center, Chapel Hill, NC

ABSTRACT

In North Carolina, crash locations are documented only with the road names for the nearest intersection. Some projects require more accurate location for importing into ArcGis. We use a combination of SAS® and GoogleEarth to more accurately pinpoint crash locations. This paper discusses how we streamline a manual process to make it feasible.

INTRODUCTION

Manually geocoding data points is both painful and time-consuming. We eagerly await the day when crash reports include a lat/long reading. Until then, what we have in North Carolina is a commonly used legacy system that locates crashes relative to the nearest intersection. Recently, some projects have needed to exactly locate pedestrian crashes. A verbal description of the crash, a diagram of the setting, and the intersection-referencing information provide the hints we need to pinpoint the crash locations. Since many pedestrians are hit in parking lots and other non-roadway locations, we need the extra value provided by Google Earth. Google Earth provides the ability to see buildings, parking lots, and other landmarks referenced in the crash narratives. We are able to match the diagrams and descriptions in the data with the photographic evidence visible in Google Earth. We use SAS to send information through various processing tools that ultimately give us a spreadsheet of XY coordinates we can import into ArcGIS.

PROCESS OVERVIEW

We use the new Google Earth API V3 key to request coordinates of the reference intersection for each crash. Using these coordinates we create a KML file containing a placemark for each crash. Viewing the KML in Google Earth, we use the crash narratives, diagrams, and data to nudge each placemark into the correct location. The resulting saved KML file is then parsed with SAS to extract the corrected coordinates and write a spreadsheet for import into ArcGIS.

REQUESTING COORDINATES FROM GOOGLE EARTH

The free Google Earth API key allows a user to request geographic coordinates for an address. There are a variety of ways to send the request. We have used the deprecated API V2 service which can provide the results in CSV format. The latest version (V3) returns the results in XML. We use V3 in this paper. Google Earth returns a plethora of information in the XML, but we are only interested in the latitude and longitude readings. Consequently, we parse the XML rather than build a SAS XMLmap.

We start with a table having one row for each crash containing the street names for the nearest intersection. We use the URL filename access method to send a geocoding request for each row in the table.

```
filename x URL "http://someplace.com/whatever";
```

First we construct the request for a URL filename from which to read the results. The required parameters are the address and the API key. We store the URL for the request, along with the API key, both of which will never change, in a variable:

```
str='http://maps.google.com/maps/api/geocode/xml&key=myAPIkey&address=';
```

The address for geocoding will vary from record to record, and should look something like this:

```
Martin Luther King at Estes Drive, Chapel Hill, NC
```

We replace the spaces with '+' for the request:

```
Martin+Luther+King+at+Estes+Drive,+Chapel+Hill,+NC
```

Then we concatenate the address variable by variable onto the end of our request string:

```
str=trim(str)||trim(left(translate(on_rd,'+',' ')))||' at ';
str=trim(str)||trim(left(translate(from_rd,'+',' ')))||' at ';
```

When we have finished, the variable str will contain the entire request string, stored as a variable in a SAS table. We can now construct a macro that will read one record from this table, send a geocoding request, parse the results, and insert a row into our final results table. We will loop through the macro, reading each record in turn.

First we build the empty results table:

```
proc sql;
create table latlong  (id num(8), on_rd char(50), from_rd char(50),
lat char(15), long char(15));
quit;
```

We get a count of intersection for the upper bound of our loop:

```
proc sql;
select count(*)  into :nrecs from intersections;
quit;
```

We begin construction of the macro. We store the Google Earth request string and other desired identifying information in macro variables:

```
%macro getaddr;
%do i=1 %to &nrecs;
data _null_;
set intersections(obs=&i firstobs=&i);
call symput('str',str);
call symput('id',id);
call symput('on_rd',on_rd);
call symput('from_rd',from_rd);
run;
```

We begin a data step that makes the geocoding request and parses the results:

```
data _null_;
filename x URL "%str(&str)";
infile x;
```

We know the latitude and longitude readings are nested within the <location></location> tags like this:

```
<location>
  <lat>35.9351960</lat>
  <lng>-79.0561319</lng>
</location>
```

This code locates and extracts the two readings, storing them in macro variables:

```
do until (index(_infile_,'<location>')>0);
input;
end;
input;
lat=scan(_infile_,-2,'<>');
call symput('lat',lat);
input;
long=scan(_infile_,-2,'<>');
call symput('long',long);
```

We needn't read anymore from the XML now that we have our information, so we quit the data step.

```
stop;
run;
```

Now we have our values to insert into the table we prepared:

```
proc sql;
insert into latlong (id,on_rd,from_rd,lat,long)
VALUES (&id,"&on_rd", "&from_rd", "&lat", "&long");
quit;
```

We have finished processing one table row, so we end the macro loop and end the macro:

```
%end;
%mend getaddr;
```

WRITING THE KML FILE

Using the information in our table named latlong, we write a KML file, which is an XML file specification used to display geographic data in Google Earth. Here is a very simple KML file for a single placemark:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml>
    <Document>
       <Placemark>
           <name>Sample Placemark</name>
           <Point>
                  <coordinates>-79.05613199999999,35.93519600000001,0</coordinates>
           </Point>
       </Placemark>
    </Document>
</kml>
```

We need the Placemark section to repeat for each intersection, or row, in our SAS table. We would like to have the road names included as descriptive data. These can be placed within a <description> tags for each placemark.

The opening tags, <?xml…>, <kml>, and <Document> are written at the beginning of the program using the _n_=1 conditional logic. Since we'll need to write closing tags, we include the end= parameter with the set statement. The altitude for each placemark will always be zero, so we assign the value 0 to the variable ht:

```
data _null_;
set latlong end=eof1;
file "SESUG2010.kml";
if _n_=1 then do;
    put '<?xml version="1.0" encoding="UTF-8"?>'
    / '<kml>'
    / '<Document>';
end;
ht=0;
```

We use this code to write a placemark for each row in the table:

```
put '<Placemark>';
put '  <name>' id '</name>';
put '  <description>';
put '<![CDATA[' on_rd 'at ' from_rd  ']]></description>'  ;
put '<Point><extrude>1</extrude><altitudeMode>relativeToGround</altitudeMode>';
put '<coordinates>' long +(-1)',' lat +(-1) ',' ht '</coordinates>';
put '</Point>';
put '</Placemark>';
```

We look for the last record in order to write the closing tags and finish our data step:

```
if eof1 then put '</Document></kml>';
run;
```

We now have a kml file to open in Google Earth.

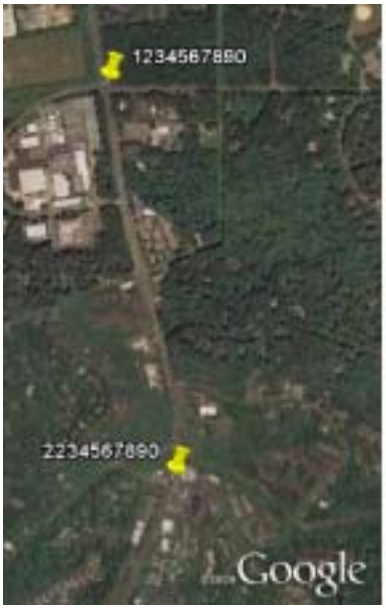

Figure 1 **Figure 2** **Figure 3**

Figure 1 reveals both placemarks, each revealing the ID for the crash. Zooming in on one of them (Figure 2), we can see the details of parking lots, buildings and other information that will help place the crash more precisely, given the information we may have for the crash. Figure 3 shows the results of clicking the placemark; this reveals the information contained in the description tag section for the placemark. Right-mouse-clicking and selecting 'properties' for a placemark puts it in a mode that allows it to be moved around using drag-and-drop. After all the placemarks have been properly positioned, we save the KML file.

FROM KML TO ARCGIS

Version 9.3 of ArcGIS allows the import of KML to a geodatabase through the Data Interoperability Extension. Absent that latest version, there are other minimal-effort possibilities. It is now possible to open XML in Excel, save a copy in .xls format, and import the spreadsheet as xy coordinates into ArcGIS. For demonstration purposes, we show how the KML can be parsed with SAS to extract the values, and written to Excel for import into ArcGIS.

We're not interested in anything in the kml file before the first placemark, so we again use _n_=1 processing logic:

```
data crashes;
infile "SESUG2010.kml" ;
if _n_=1 then do until (index(_infile_,"<Placemark>")>0);
    input ;
end;
```

Now we begin to look for the first line containing information we need, which is the <name> tag. The crash identifier is found between the tags. Here we show the KML followed by the SAS code to extract the id value:

```
<name>1234567890 </name>
```

```
do until (index(_infile_,"<name>")>0);
    input;
end;
id=input(scan(_infile_,-2,'<>'),10.);
```

Coordinates appear after the name. We extract the two coordinate values from the KML text and output the record:

```
do until (index(_infile_,"<coordinates>")>0);
input ; end;
lat=input(scan(_infile_,-3,'<>,'),10.7);
long=input(scan(_infile_,-4,'<>,'),11.7);
output;
run;
```

We now have a SAS table containing the corrected latitude and longitude. Absent SAS Bridge to ESRI, we write an Excel file for import into ArcGIS as XY values:

```
libname a "crashes.xls";
data a.crashes;
set crashes;
run;
libname a clear;
```

CONCLUSION

We have demonstrated a process to streamline a very time-consuming but valuable task. Being able to exactly place the crashes makes it possible to determine the most common types of pedestrian crashes, leading to potential treatments and other countermeasures that could save lives.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Carol Martell
Highway Safety Research Center
730 Martin Luther King Jr Blvd
Chapel Hill, NC 27514
carol_martell@unc.edu

Daniel Levitt
Highway Safety Research Center
730 Martin Luther King Jr Blvd
Chapel Hill, NC 27514
dlevitt@hsrc.unc.edu