5 - DESARROLLO DE LAS FUNCIONES

5.1 - Toolbox (caja de herramientas) de Google Earth

Se trata de un conjunto de funciones de MATLAB con el que se puede visualizar espacialmente y temporalmente los datos distribuidos dentro de Google Earth. En conjunto, estas funciones se conocen como la Caja de herramientas de Google Earth.

Se puede discutir que la conversión de los resultados de los modelos y las mediciones para poder verlos en Google Earth es innecesaria, ya que también se puede ver en MATLAB por sí mismos. Sin embargo, usando Google Earth para la evaluación de los datos puede tener algunas ventajas importantes, algunas de las cuales se enumeran a continuación:

• Herramientas muy intuitivas para navegar de manera visual.
• Recuperación de datos de objetos en el visor.
• Reproducción rápida de archivos de gran tamaño.
• Elección de forma interactiva los objetos que se debe mostrar.
• Representación dinámica de datos variables-tiempo.

5.1.1 - Configuración de Google Earth

Para evitar problemas con la visualización e interpretación de los archivos de Google Earth, puede ser útil contar con su copia de Google Earth creado de acuerdo con las siguientes especificaciones:

• Haga clic en 'Herramientas' en el menú de Google Earth y seleccione "Opciones...". En la ficha denominada '3 D View 'comprobar que "Modo de gráficos" está establecido en "OpenGL".
• En la misma ficha, establecer el formato latitud / longitud para "grados" en "Mostrar lat. / long.
• También en esta ficha, establecer las unidades de elevación a 'Metros' en ' Elevation Show'.

5.1.2 - KML: el lenguaje de Google Earth

La gran flexibilidad de Google Earth se deriva de su uso de archivos de texto basados en XML, conocido como archivos KML. Este tipo de archivo por lo general contiene una serie de objetos, tales como polígonos, líneas y puntos. Cada uno de estos objetos se representa dentro de un archivo KML por sus etiquetas KML. Por ejemplo, podemos encontrar un objeto polígono:

<Polygon>
...
</ Polygon>

O una línea de objetos:

<LinearRing>
...
</ LinearRing>

Propiedades tales como line width, coordinates, y polygon color se pueden especificar con el uso de etiquetas adicionales. A una línea se le puede asignar un estilo de línea de la siguiente manera:

<LineStyle>
  <color> ffffffff </ color>
  <width> 1,00 </ width>
</ LineStyle>

Lo que representa una opacidad total, la línea blanca del ancho de 1.

Por supuesto, la escritura de etiquetas KML manualmente en un editor de texto no es práctico cuando se trabaja con conjuntos de datos que contiene más de unos pocos objetos. Afortunadamente, sin embargo, la tarea de escribir estos archivos puede ser totalmente automatizada con la Caja de herramientas de Google Earth.

5.1.3 - Estructura de la caja de herramientas

El núcleo de la caja de herramientas está formado por un grupo de aproximadamente 20 funciones de MATLAB (archivos *. m), ubicado en la carpeta raíz caja de herramientas ("googleearth / '). La mayoría de estas funciones generan matrices de caracteres en concordancia con el estándar KML.

Para cada uno de los m-files, un archivo de ayuda se ha incluido que se puede ver desde el navegador de ayuda de MATLAB. Los archivos de ayuda se encuentran en "googleearth / html '. Además de los archivos de ayuda, las manifestaciones han sido incluidos ("googleearth / demo). Muchas de estas demostraciones de escribir su salida en una carpeta separada ("googleearth / kml") con el fin de evitar la contaminación de los directorios con los archivos que no deberían estar allí.

La carpeta "Googleearth / tutres ' contiene archivos de recursos usados en este tutorial.

La carpeta "Googleearth / datos" contiene archivos adicionales, tales como las imágenes de iconos y modelos Collada.

"Googleearth / doc ' contiene los archivos relativos a la documentación para imprimir, por ejemplo, el presente documento.

Por último, una carpeta 'googleearth / tmp' se ha incluido, que es utilizado por algunas de las funciones para la escritura de datos temporales. El contenido de esta carpeta se vacía automáticamente por algunas funciones, por lo tanto, no debe guardar datos importantes en ella.

5.1.4 - Adición de la caja de herramientas a matlab

Para poder utilizar las funciones situado en la caja de herramientas, primero debe especificar el lugar donde se encuentra la caja de herramientas, por ejemplo:

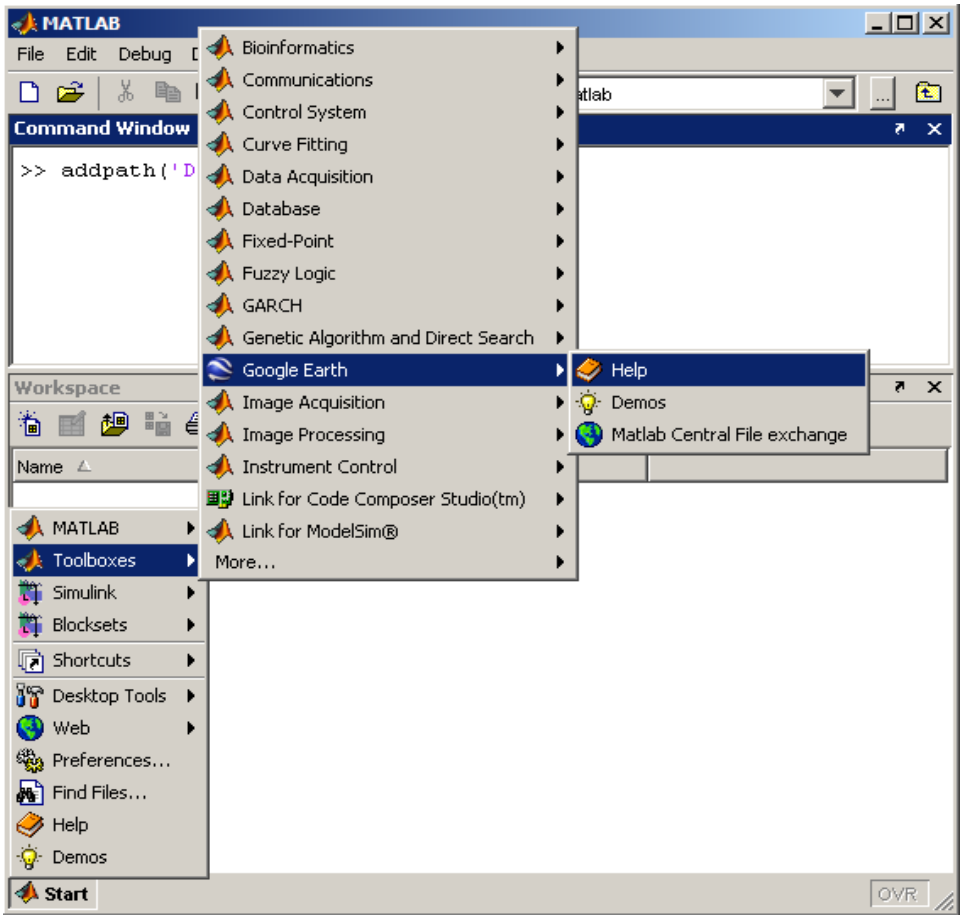
>> Addpath ('D: \ matlab \ GoogleEarth)

Esto agregará el directorio indicado a la ruta de búsqueda de MATLAB .

5.1.5 - Acceso a la documentación

La caja de herramientas ha sido diseñado para ser lo más intuitiva posible. Para ello, la sintaxis de las funciones es muy similar a sus homólogos de MATLAB. Además de eso, un archivo de ayuda está disponible para todas las funciones dentro de la caja de herramientas. En estos archivos, usted puede encontrar información sobre lo que hace una función en particular, cuáles son las variables que se pasan, las opciones de la función, enlaces a otras funciones que realizan tareas relacionadas, y un ejemplo de cómo se puede utilizar la función.

Haga clic en el botón "Inicio" que se encuentra en la esquina inferior izquierda de la aplicación de MATLAB y seleccione "cajas de herramientas". La Caja de herramientas de Google Earth debe estar presente (ver imagen abajo).



Puede acceder a la documentación mediante el comando siguiente en el símbolo del sistema:

>> Ayuda googleearth

Haga clic en el vínculo "Contenidos de la caja de herramientas de Google Earth" para ir a la lista alfabética de las funciones que en conjunto constituyen la Caja de herramientas de Google Earth.

Una vez explicado de manera aproximada en qué consiste la caja de herramientas en la que nos apoyaremos para el desarrollo nuestras funciones, vamos a mostrar el desarrollo de las distintas funciones que hemos creado, en qué consisten cada una de ellas, los pros y los contras de esa representación, y las mejoras que creemos que podemos añadir, con lo que tendremos una gran variedad de funciones creadas, permitiendo a cada uno decidir cuál cree que es la más conveniente para representar según que variable.

5.2 - Desarrollo de la función para dibujar círculos

Inicialmente pensamos en crear una representación que nos aportara una serie de círculos a lo largo del camino que representaran el valor de los datos. La serie de

círculos estarían colocados a lo largo de la trayectoria, y lo que representaría el valor de la variable seria el diámetro de este.

Una vez pensado, desarrollamos la función primera que aportamos en el anexo. Esta función, como ya hemos comentado, está apoyada en la Caja de Herramientas de google earth que se ha explicado antes. Iremos explicando las partes principales para la comprensión de la función y sobre todo lo necesario para su utilización.

Inicialmente nos encontramos con la definición de los vectores que vamos utilizar en la función como podemos ver en el extracto de código que adjuntamos a continuación:

```
X = data(:,4); % segunda columna
Y= data(:,3); % tercer columna
R = data(:,7); % velocidad

numeroelem=length(data);
```

Podemos comprobar que tendremos tres vectores, uno de ellos será la longitud, otro la latitud (X e Y respectivamente), y finalmente un tercer vector, que en principio puede ser cualquier variable, y que nosotros en el ejemplo la hemos colocado como la velocidad.

Se puede observar también que las columnas las extraemos de la matriz data, que ya explicamos cómo introducirla en el workspace de matlab, y que los números de las columnas tienen que coincidir con la columna de correspondiente de la matriz, es decir, en el vector R = data(:,7) hemos elegido la columna 7 de la matriz data, que representara una variable determinada, la cual podremos ir variando simplemente cambiando el numero de la columna según que variable queramos representar.

Lo que nos aporta numeroelem es el número de elementos de los vectores que acabamos de definir.

A continuación lo que haremos es crear el vector kmlStr con la ayuda de las funciones de la caja de herramientas de googleerth:

```
kmlStr = [];

for i = 1:50:numeroelem % cada 50 puntos
   kmlStr = [kmlStr,ge_circle(X(i),Y(i),R(i)*10)];
end
```

Podemos observar tres cosas distintivas, la primera, que dentro de kmlStr llamamos a la función ge_circle, que es una de las comentadas funciones de la caja de herramientas de Google Earth. A esta función, lo que tenemos que pasarle, son los tres vectores de datos antes creados, imprescindible la latitud y la longitud, y la tercera variable que será la que queremos representar a lo largo del recorrido, en nuestro caso como ya hemos comentado, la velocidad.

La segunda cosa es como varían los valores en nuestro bucle “for”, tenemos puesto un 50, que lo que indicara que solo se cojan los valores de 50 en 50 en nuestros vectores, lo cual lo podemos cambiar simplemente cambiando este valor por otro que nosotros queramos, esto lo variaremos a continuación para mostrar la diferencia de coger más o menos valores.

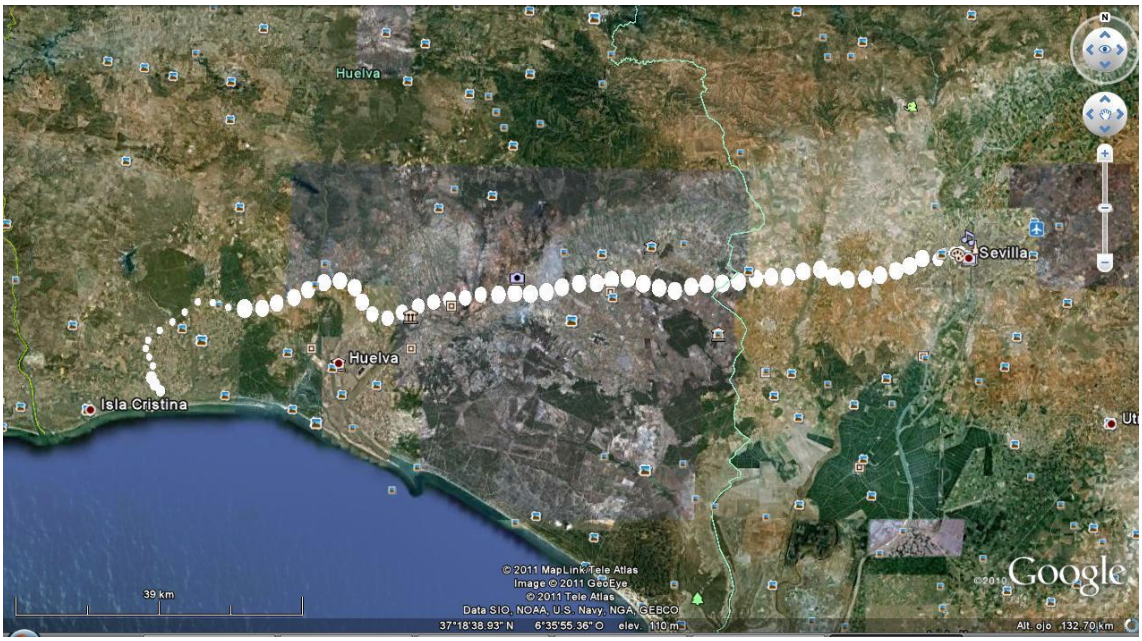
La última observación es el valor que multiplica al valor R(i) cuando llamamos a la función ge-circle, esto simplemente está dirigido a que se hagan mayores todos los valores y por tanto se vean mejo, esto lo mostraremos a continuación con una imágenes.

Finalmente con la función ge_output de nuestra caja de herramientas de G.E. creamos el archivo .kml para google earth. Este archivo tal y como lo tenemos puesto en el código, se creara en la misma carpeta donde tengamos el programa, es decir, en este caso, en la carpeta donde tenemos el programa de creación de círculos, que vamos a llamar “dibujarcirculos.m”.

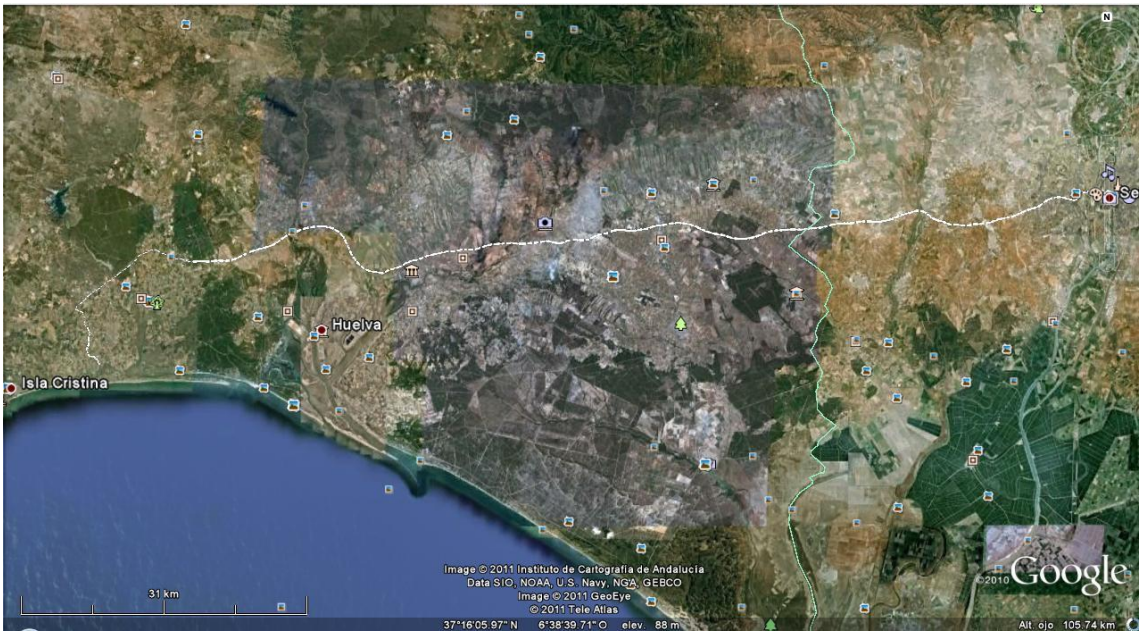
```
kmlFileName = 'obd_circle.kml';
kmlTargetDir = [''];%..',filesep,'kml',filesep];

ge_output([kmlTargetDir,kmlFileName], [kmlStr],'name',kmlFileName)
```

kmlFileName lo único que nos va a proporcionar es el nombre del archivo kml que se va a crear.

A continuación veamos unas imágenes del archivo creado, en este primer caso cuando el programa lo corremos tal y como está en la explicación, es decir, cogiendo de 50 en 50 los elementos y multiplicando por 10 el valor de la velocidad:



Como podemos observar se trata de un trayecto desde Sevilla hasta Isla Cristina, donde podemos ver aproximadamente que velocidad se ha llevado, y el trayecto que se ha recorrido. Pero podemos comprobar que de esta manera se puede ver mejor el conjunto del trayecto completo, pero es menos preciso. A continuación se muestra el mismo trayecto pero cogiendo los datos de 5 en 5 elementos y sin amplificar las velocidades:



Comprobamos como la precisión del recorrido es mucho mayor y la localización de posibles eventos sería mucho más precisa, pero tenemos que acercarnos mucho más en el mapa para poder apreciar bien la representación.

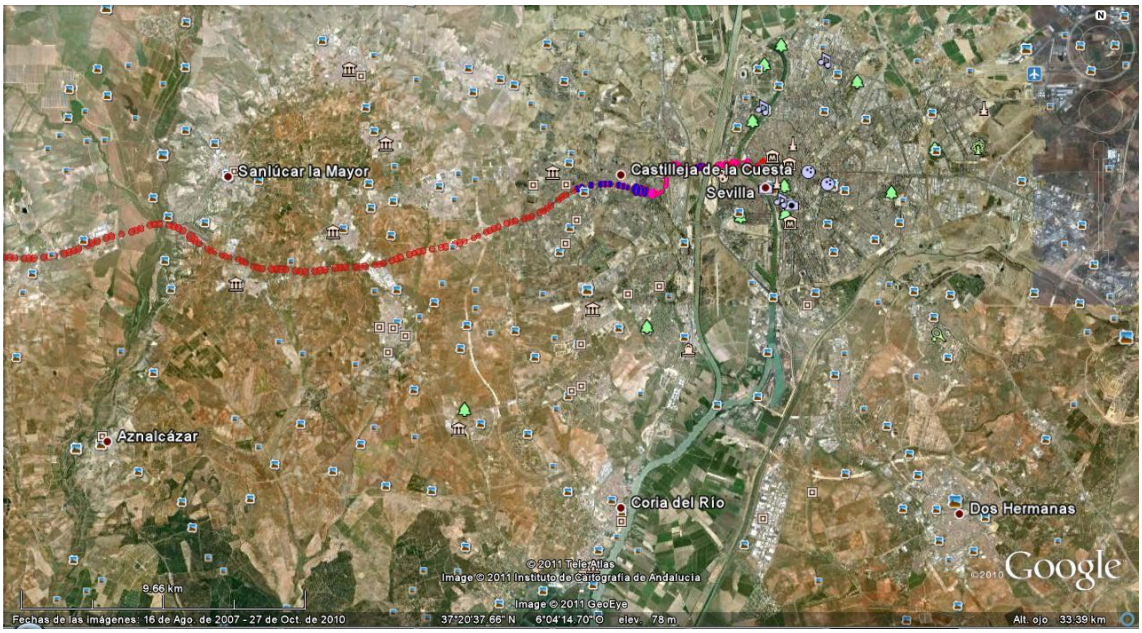
Viendo todo esto, se decide, para mitigar el problema de la difícil localización de eventos cuando los datos tomados son muchos, se piensa en poner de distintos colores los círculos según el valor de las velocidades lo que permite evaluar las velocidades con mayor facilidad no solo por la capacidad de verlos todos directamente desde una vista más alta, sino por el simple hecho que la visualización de colores es mas intuitiva y simple que la de los tamaños de los círculos.

Se procede por tanto a incluir en este programa unas opciones de la siguiente manera:

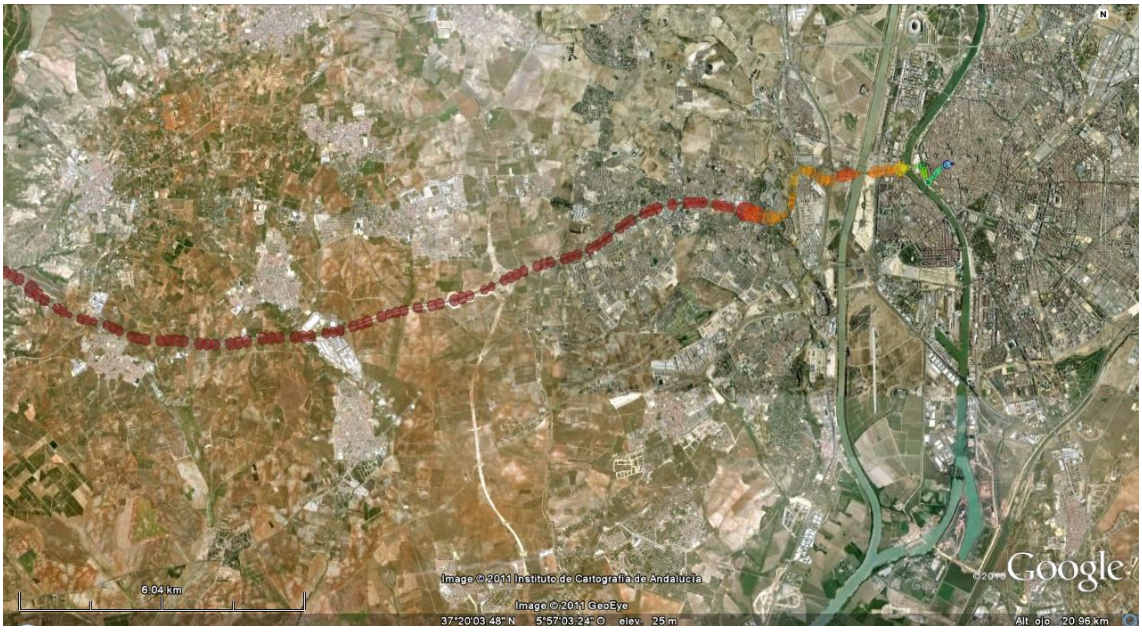
```
for i = 1:5:numeroelem % cada 5 puntos

    if V(i)<20
        colorcirculo='00FF0000';
    elseif V(i)<50
        colorcirculo='FFFF00EE'
    elseif V(i)<90
        colorcirculo='FFFF00EE';
    elseif V(i)<120
        colorcirculo='EE0000EE';
    else
        colorcirculo='80FF0000';
    end


   kmlStr = [kmlStr,ge_circle(X(i),Y(i),RPM(i)/20,... % revoluciones
dividido entre 2 para escalar
                           'lineColor','FFFF0000', ...
                    'polyColor',colorcirculo)];
end
```

La función ge_circle tiene (al igual que las demás que vamos a usar) tiene unas opciones entre las que se encuentra la de colorear los círculos que hemos creado, por ello, y usando el bucle “for”, creamos una variable llamada “colorcirculo”, que dependiendo del valor de la variable, tomara un valor u otro.

En el caso de nuestro programa aquí puesto vamos a representar dos variables, una como la velocidad que serán los códigos de colores, y el diámetro como las revoluciones, aunque veremos que resulta un poco difícil de interpretar los dos a la vez. Este programa lo llamaremos “dibujacirculos2.m”.



Lógicamente para una mejor visualización podemos colocar una mayor cantidad de códigos de colores y reducir el intervalo de valores de la velocidad para cada color lo que quedaría de la siguiente manera (el código se adjunta en el anexo)con el programa “dibujarcirculos2a.m”:



Lo cual nos da una representación mucho más clara de los valores que estamos representando.

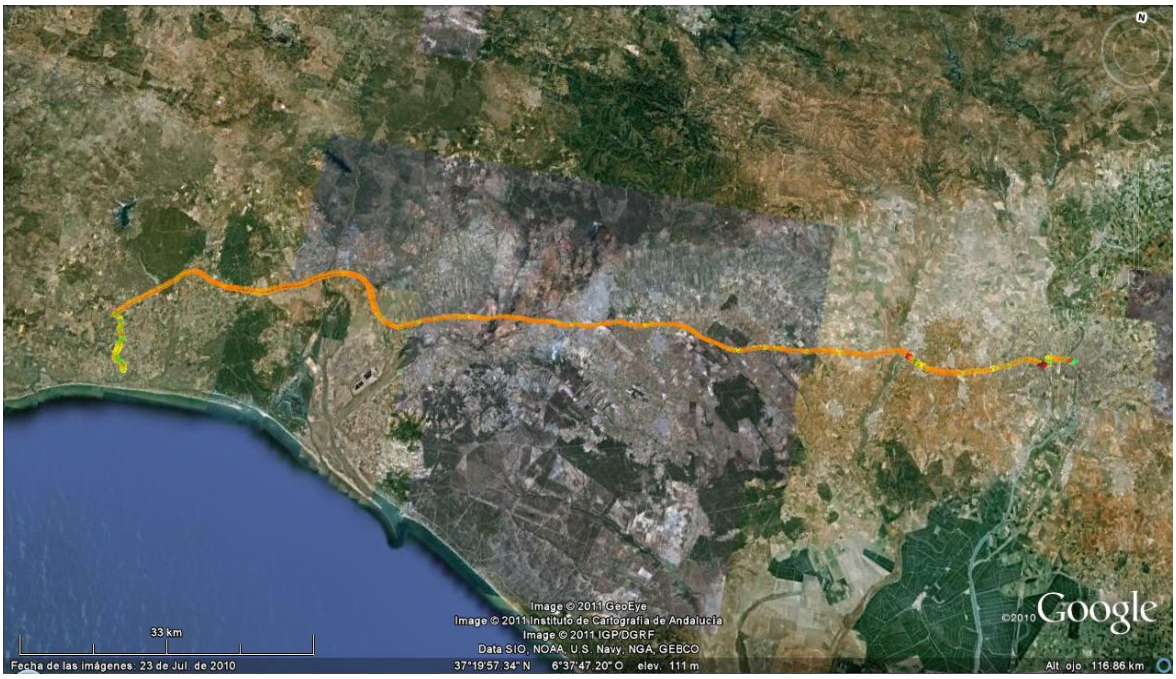
Finalmente de este programa queda señalar que la representación de más de una variable (aparte lógicamente de la latitud y la longitud) es bastante poco intuitiva, por lo que se recomienda la representación de una sola.

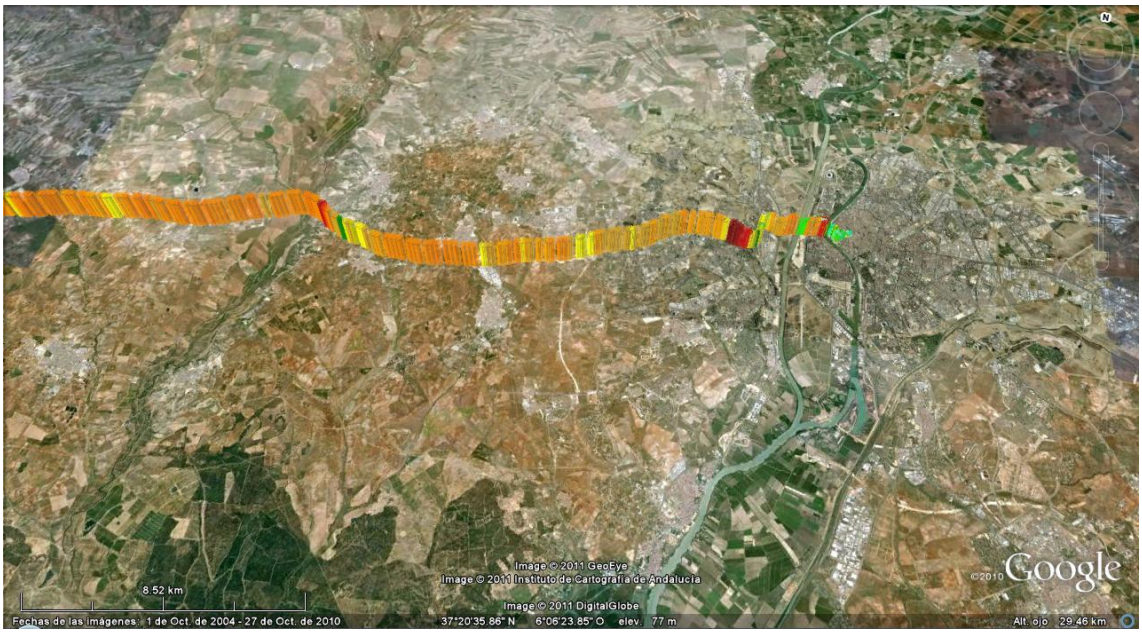
5.3 - Desarrollo de la función para crear cilindros

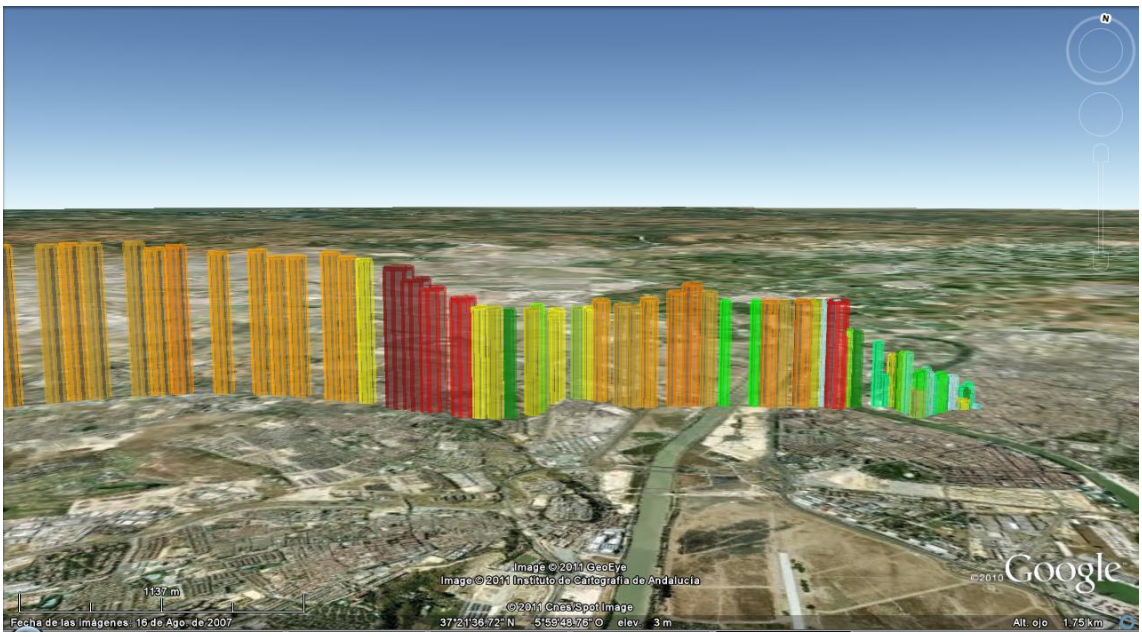
Una vez desarrollada la función que representa los datos como círculos de distintos diámetros y también de distintos colores, pensamos que podríamos representar estas variables como cilindros de distintas alturas y distintos diámetros. Emplearemos para ello la función ge_cylinder.

Como en el caso de los círculos, los diámetros son bastante poco claros a la hora de interpretar los valores, así que una vez representamos los cilindros sin una escala de colores, pensamos que deberíamos hacerlo, colocando la variable que se representa con el diámetro del cilindro representada también por los colores.

El programa es muy parecido al anterior, como podemos observar en el anexo, solo variando las opciones, la función a la que llamamos y las variables que introducimos. Se denominara “dibujar_cilindro.m”. Se obtienen los siguientes resultados corriendo el programa y representando revoluciones y velocidades:







Podemos observar que el recorrido es el mismo que el elegido anteriormente, Sevilla – Isla Cristina, donde se puede ver con bastante claridad la representación de las variables. Aunque la representación de las dos variables es bastante más clara que en el caso de los círculos, sigue sin ser intuitiva del todo, por lo que se dejara a decisión del usuario la elección de representar una o dos variables según considere.

5.4 - Desarrollo de la función para crear rectángulos

Una vez creadas las funciones anteriores, comprobamos la dificultad de la interpretación de los valores de las variables según el radio tanto de los círculos como de los cilindros, que también acarrean una discontinuidad necesaria en la representación de los datos debido a la inevitable necesidad que los citados círculos y cilindros no se superpongan entre ellos.

Para evitar esto y hacer no solo más clara, sino también más precisa la interpretación de los datos, se procede a continuación al desarrollo de una función que nos represente las variables en forma de rectángulos continuos, cuya altura representara el valor de la variable, el color el valor de otra posible variable o de la misma a elección, y el ancho de cada rectángulo dependerá del intervalo de cogida de datos del vector que se nos aporta, pudiendo ser en este caso de uno en uno, y no teniendo necesariamente que hacerlo como en los otros programas, que hemos tenido que cogerlos como ejemplo de 5 en 5 y de 50 en 50 (no podemos cogerlos de uno en uno por que la representación de los diámetros sería tan pequeña que se volvería aun más complejo interpretar los valores de lo que ya es).

El programa desarrollado se llamara “dibujarectangulos.m”, y en esencia realiza las mismas operaciones que los programas anteriores solo variando en lo que vamos a comentar a continuación.

En primer lugar varia la forma de los vectores que le pasamos a la función de llamada que en este caso será “ge_poly3”. Los vectores los tomamos de la siguiente manera:

```
X = data(1:5:NN,2); %[x,fliplr(x)];
Y = data(1:5:NN,3); %[ones(size(x))*lat,ones(size(x))*lat];
Rev = data(1:5:NN,5); %[ones(size(x))*e1,ones(size(x))*e2];
Vel = data(1:5:NN,4);
N = size(X,1)

x=[X'];
y=[Y'];
m=zeros(1,N);
z=[m,Vel'];

%Initialize kml string:
kmlStr= '';

for i=1:1:N-1
    %Shift polygon by j degrees longitude:
   sx = [x(i),x(i+1),x(i+1),x(i)];
   sy=[y(i),y(i+1),y(i+1),y(i)];
   sz=[z(i),z(i+1),z(N+i+1),z(N+i)];
```

Como podemos comprobar los primeros vectores, es decir, X, Y, Rev y Vel son exactamente iguales a los que tomamos en las funciones anteriores, pero la diferencia radica a partir de ahí. En este caso al estar representando el rectángulo, no solo tenemos que aportarle a la función de llamada el valor de la variable en ese momento, sino que tenemos que aportarle un vector con los cuatro vértices del rectángulo, es decir, dos con valor cero (para que este pegado a tierra que generalmente es lo que querremos) uno en el elemento i del vector y otro en el i+1, y otros dos en estos mismos instantes con el valor, ya si, de la variable en cada uno de ellos. Eso se hace como viene en las líneas arriba puestas.

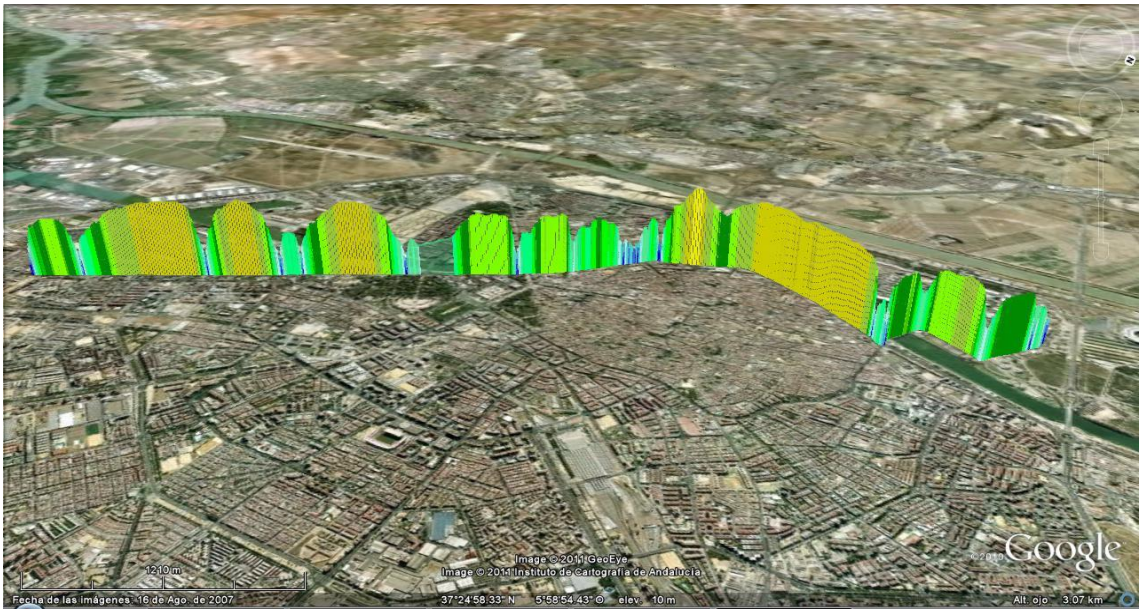
Una vez creados los vectores de cuatro elementos, un vector para la latitud, otro para la longitud y el último para la variable elegida, procedemos como en los otros casos a programar la escala de colores que definirán, en este caso la misma variable que representamos con la altura de los rectángulos. Se puede ver perfectamente en el código aportado en el anexo.

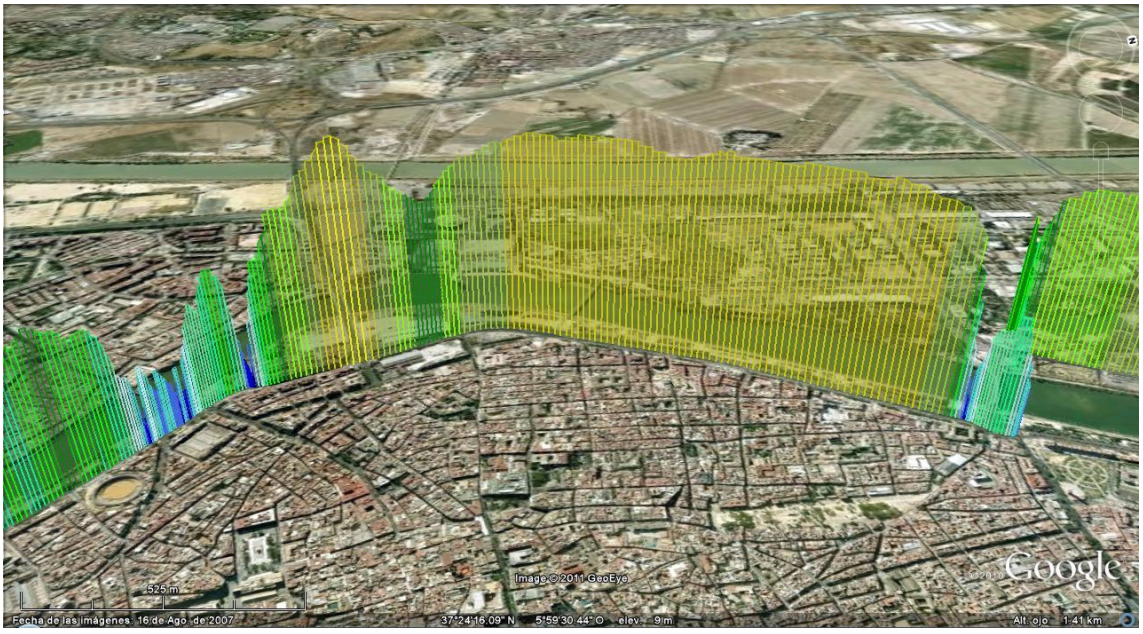
Finalmente realizamos la llamada a la función de la misma manera que lo hacíamos en los otros casos:

```
kmlStr = [kmlStr,ge_poly3(sx,sy,sz*10,...
                          'lineWidth',2.0,...
                          'altitudeMode','absolute',...
                          'lineColor',colorlinea,...
                          'polyColor',colorrectangulo,...
                          'msgToScreen',true)];


end


%ge_output('demo_ge_poly3.kml',[kmlStr]);
kmlFileName = 'grafrectangulos.kml';
kmlTargetDir = [''];%..',filesep,'kml',filesep];
ge_output([kmlTargetDir,kmlFileName],[kmlStr],...
                                        'name',kmlFileName);
```

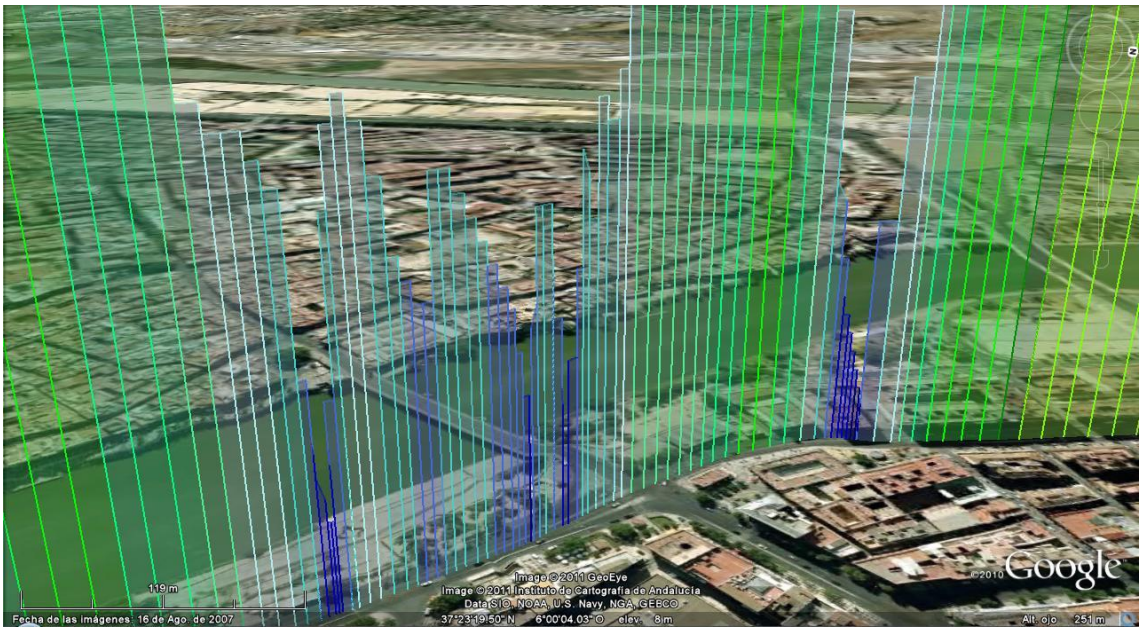
A continuación vamos a mostrar representaciones con esta función de distintas variables en distintos trayectos:

VELOCIDAD: “rectang_veloc.m”



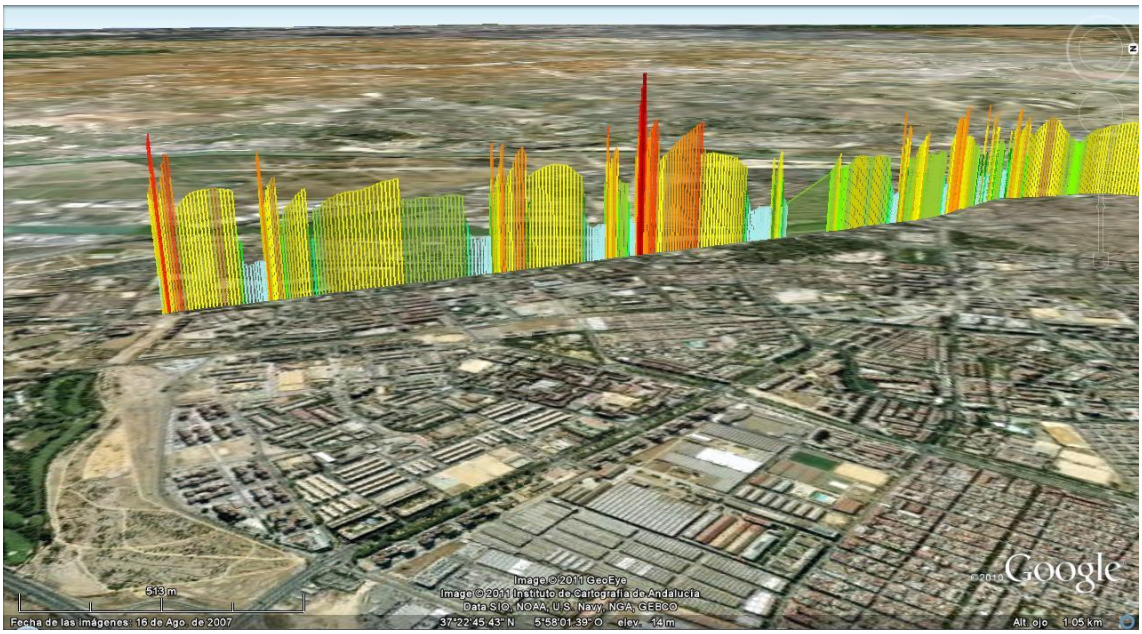


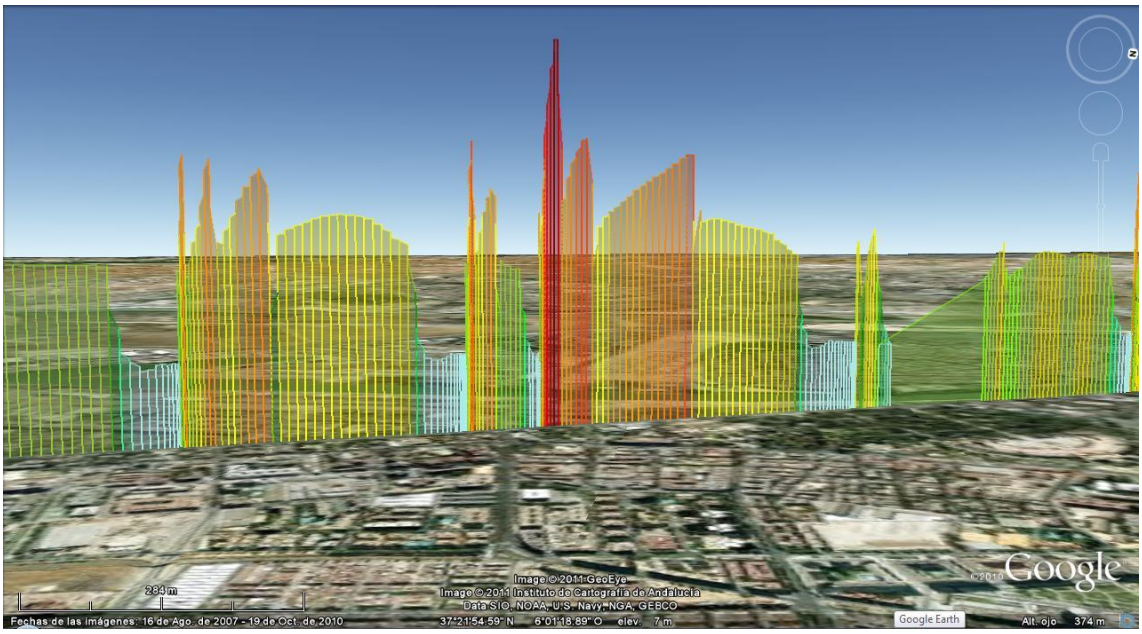
Podemos observar que es bastante más fácil la interpretación de los valores de las variables y la representación es muy clara debido a la continuidad que aporta la representación mediante rectángulos. La precisión puede aumentar si en vez de coger los datos de 5 en 5 como hemos hecho, lo hacemos de uno en uno:



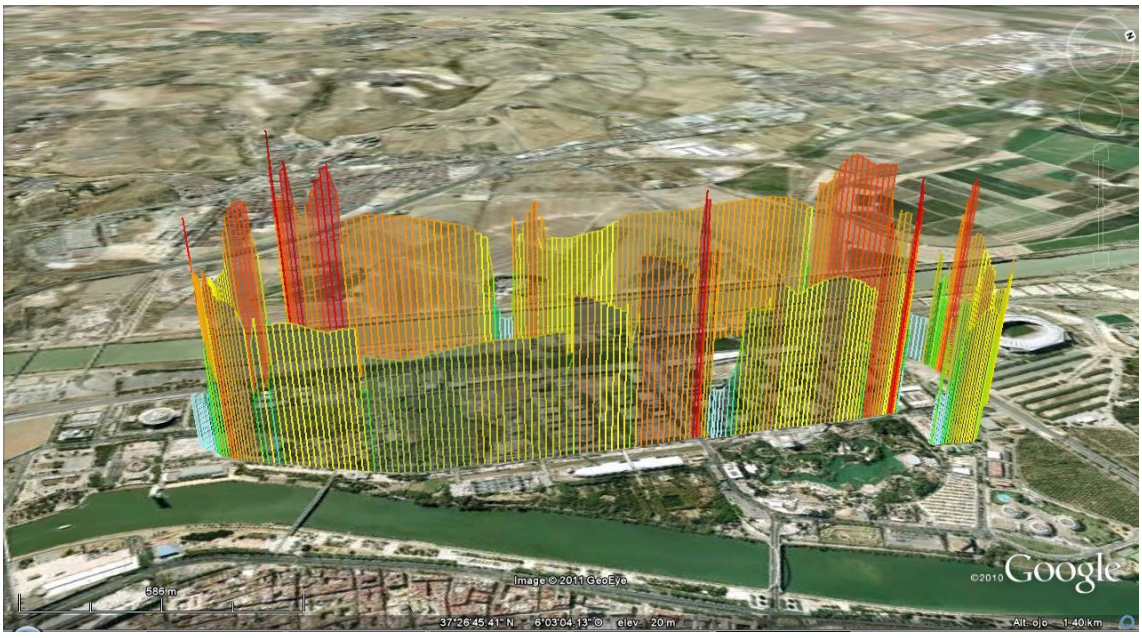
Se puede ver la gran precisión de la representación.

REVOLUCIONES: "rectang_revoluciones.m"



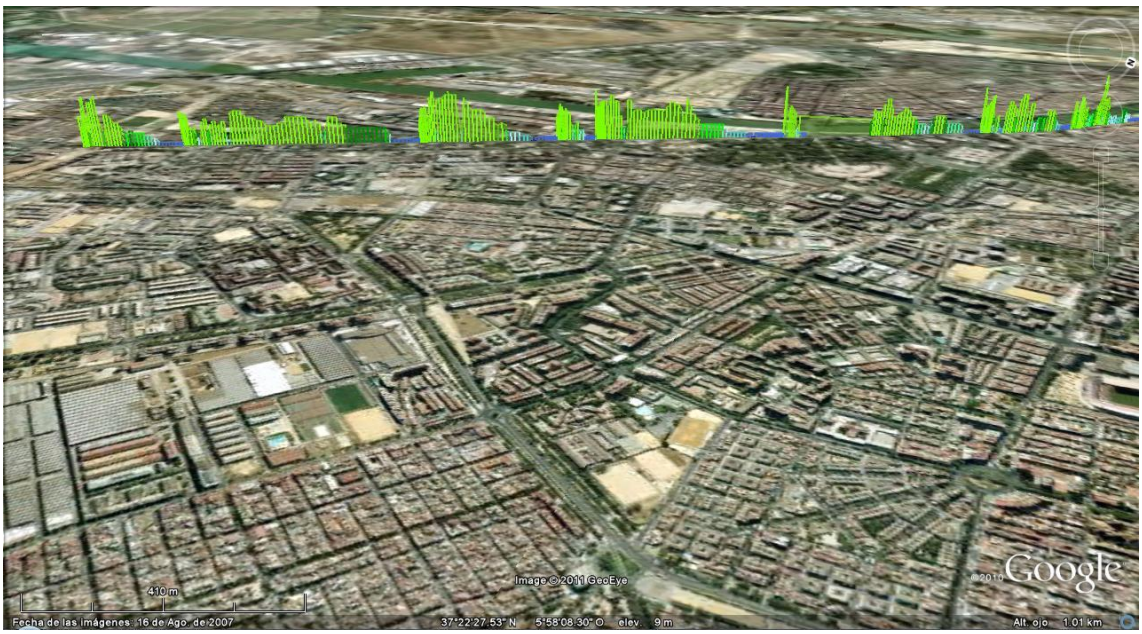


Si buscamos un trayecto distinto:



Como último ejemplo vamos a añadir la variable “Load”, que nos va a mostrar el porcentaje de potencia que estamos usando del coche.

LOAD: “rectang_load.m”



Como hemos podido observar con estos ejemplos, la representación es muy intuitiva y bastante mas precisa que las anteriores por lo que a continuacion representaremos la potencia y la relación de cambio (lo que nos indicara la marcha que metemos), con la función de representación de rectángulos.

5.4.1 - Representación de la potencia con la función de rectángulos

Una magnitud que consideramos muy importante en el momento del análisis es la potencia que emplea el coche en cada momento. Al no darnos este valor el programa de toma de datos del coche, debemos realizar una aproximación lo mas real posible de la misma, ya que obtener la potencia exacta es muy complejo y tampoco necesitamos una exactitud enorme.

Para esta aproximación simplemente hemos tenido en cuenta la siguiente formula de la potencia:

$$P = F * v$$

Siendo “v” la velocidad que lleva el coche en m/s, y “F” la fuerza en Newtons, que la podremos calcular con la formula que ponemos a continuación:

$$F = (\frac{1}{2} * v^2 * \rho_a * A * C_d) + (m * LA) + (m * g * C_r)$$

Donde:

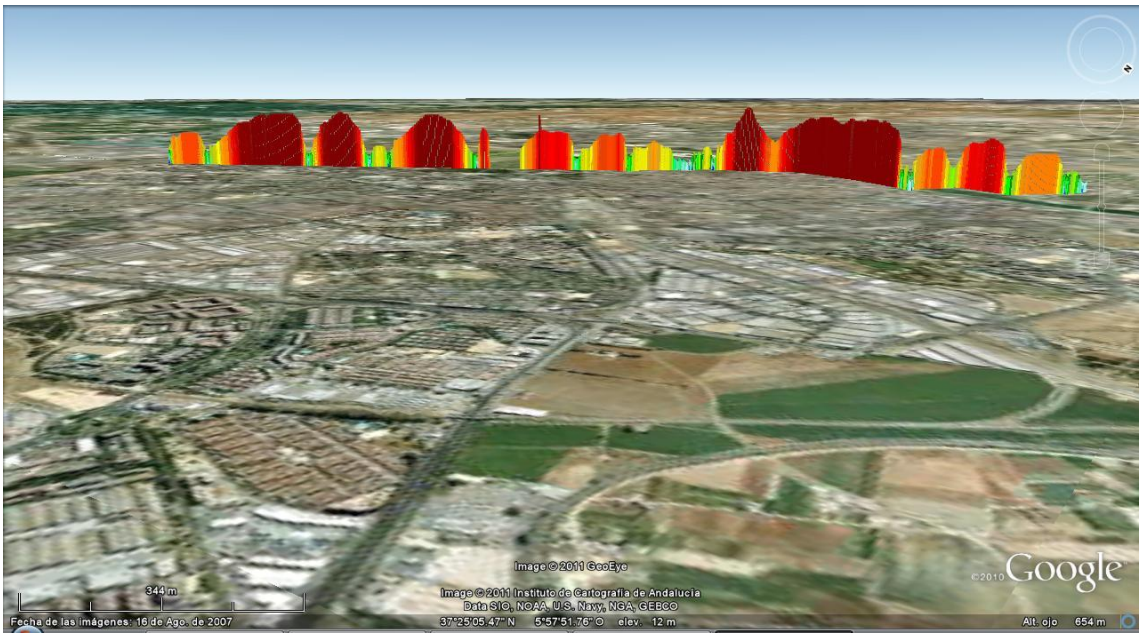
$$\rho_a \equiv densidad\ del\ aire$$
$$A \equiv Area\ frontal\ del\ coche$$
$$C_d \equiv coeficiente\ aerodinamico$$
$$m \equiv masa\ del\ vehiculo$$
$$LA \equiv aceleracion\ longitudinal$$
$$g \equiv gravedad\ (9.81\ m/s^2)$$
$$C_r \equiv coeficiente\ de\ rodadura$$

Con la combinación de estas dos funciones obtenemos la potencia que desarrolla el coche, pudiéndose ver en el programa (por los comentarios puestos a la derecha de cada vector) las unidades en las que obtenemos dicha potencia. A continuación mostramos parte del código donde se han desarrollado las formulas antes puestas y donde vienen las unidades de la potencia:

```
cr=0.015;
m=940; %peso(Kg)
g=9.8; %gravedad (m/s^2)
dens=1.2; %densidad aire (Kg/m^3)
A=2.421; %area frontal(m^2)
cd=0.325;  %drag coefficient
X = data(1:5:NN,8); %[x,fliplr(x)];
Y = data(1:5:NN,7);%[ones(size(x))*lat,ones(size(x))*lat];
vel=(1000*data(1:5:NN,6))/3600; %velocidad(m/s)
LA=data(1:5:NN,3);
N=size(vel,1)
for i=1:N
    if LA(i)<0
        LA(i)=0;
    else
        LA(i)=LA(i);
    end
end
F=((vel.^2).*dens*A*(cd/2))+(m*g*cr)+(LA.*m);
Pkw=(F.*vel)/1000; %potencia (KW)
Pcv=(1.3596*(F.*vel))/1000; %potencia (CV)
```

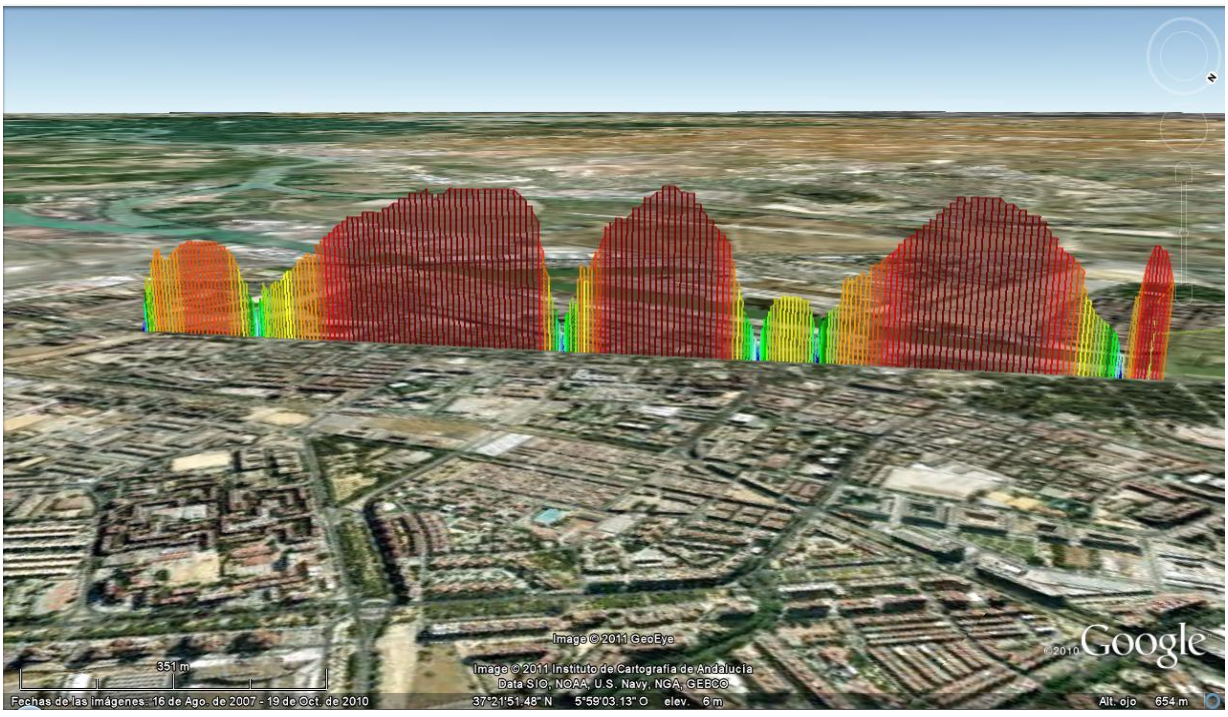
Como podemos imaginar, cada coche tendrá en valor distinto de la masa, el área frontal y el coeficiente aerodinámico por lo que en cada caso deberemos cambiar dichos valores.

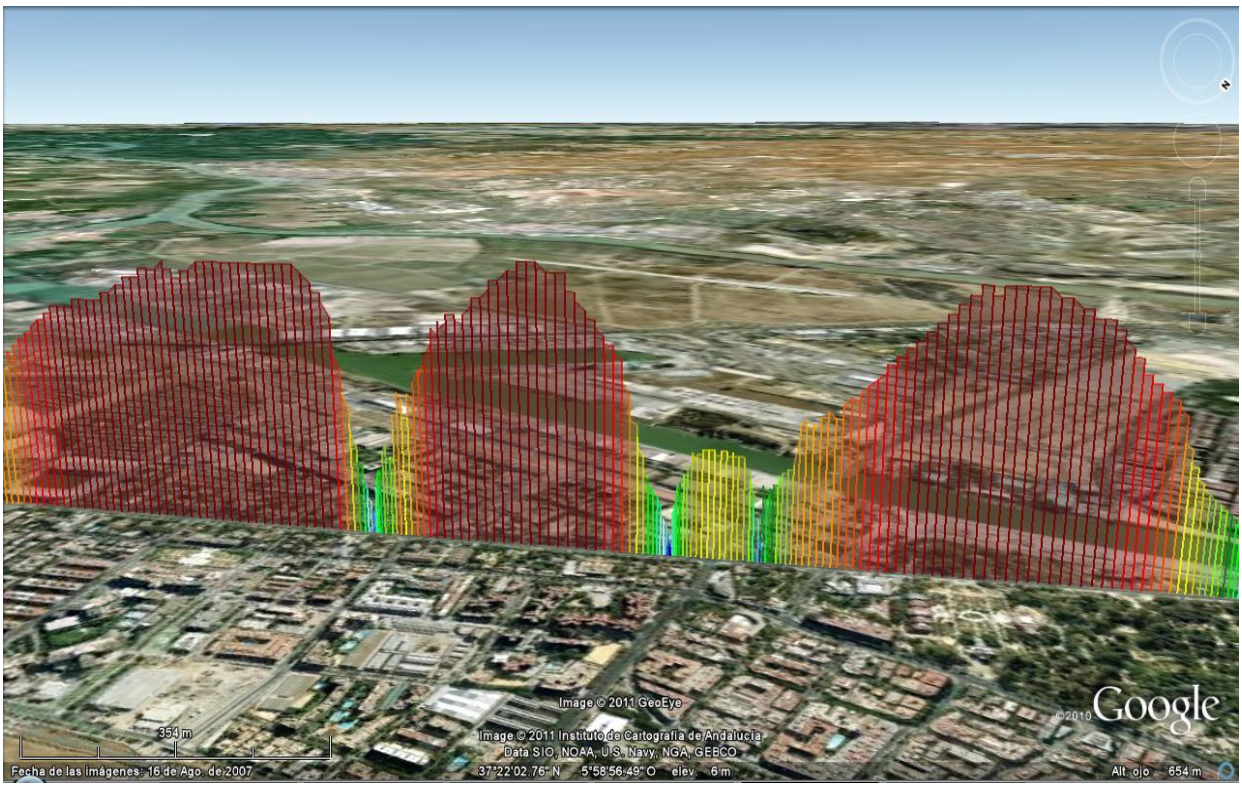
El resto del programa se puede ver en el anexo con los códigos. Mostramos a continuación resultados del programa desarrollado:



Podemos comprobar zonas de mayor potencia desarrollada, que coincidirán con las zonas de mayor velocidad, y otras con una potencia casi nula, que serán zonas donde el vehículo se ha tenido que parar. También podemos ver pendientes mas inclinadas, lo que muestra una aceleración (o deceleración) mayor, y otras de pendiente casi nula, lo que demuestra una zona de aceleración prácticamente nula.

Lo comentado se puede observar mejor en las imágenes que siguen:





5.4.2 - Representación de la relación de transmisión con la función de rectángulos

La relación de transmisión de una marcha es un valor fijo para cada una de ellas y para cada coche. La relación comentada es:

$$C_m = \frac{v}{rpm} * 1000$$

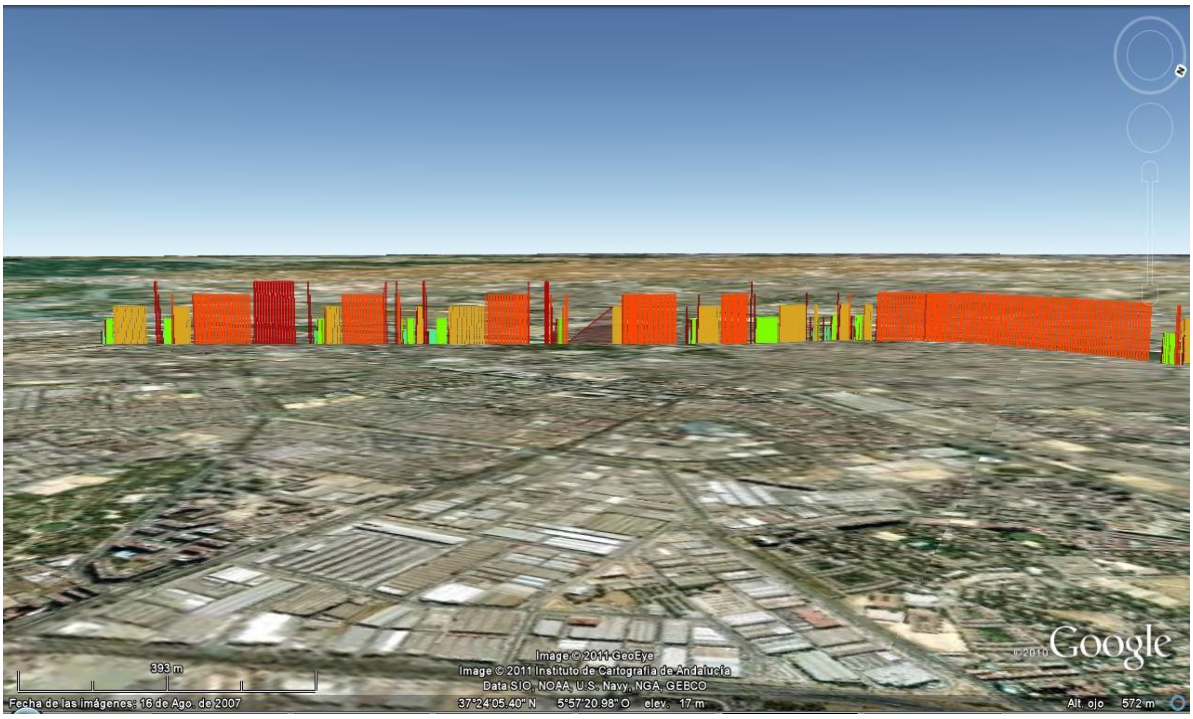
Como para cada vehículo el valor es distinto, nosotros tomaremos los del vehículo empleado, es decir, el fiat…… los valores son:

$$C_{m1} = 7.9$$
$$C_{m2} = 14.4$$
$$C_{m3} = 21$$
$$C_{m4} = 27.7$$
$$C_{m5} = 34.6$$

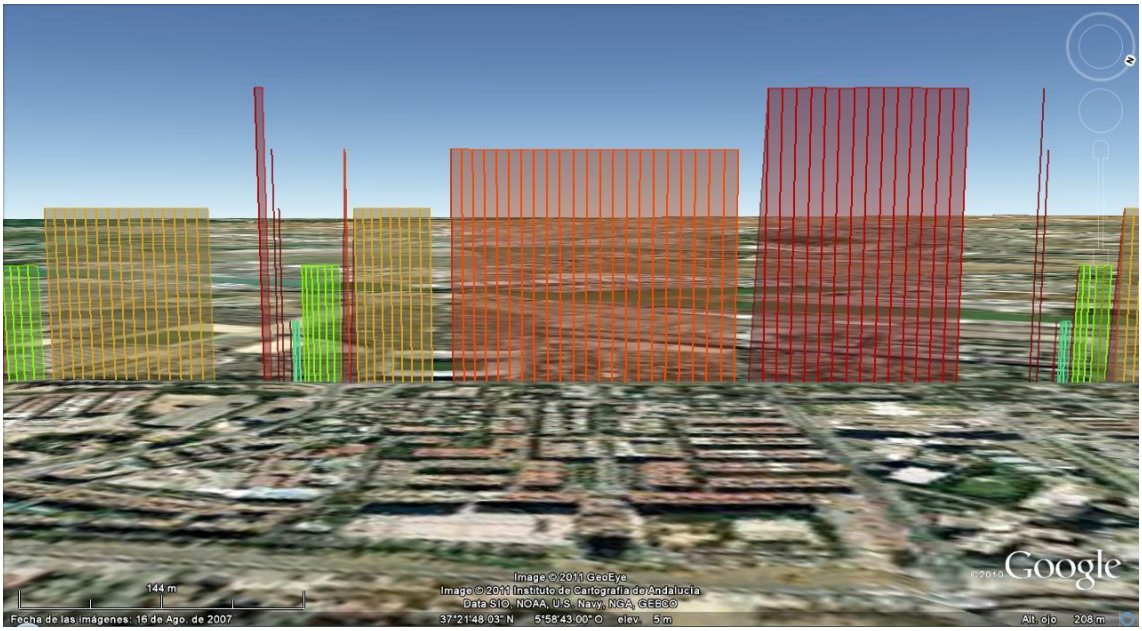
Con estos valores, vamos a realizar la función. Aportamos parte del código donde se ve la formación del vector de la relación de transmisión:

```
NN=size(data,1)
X = data(1:5:NN,8); %[x,fliplr(x)];
Y = data(1:5:NN,7);%[ones(size(x))*lat,ones(size(x))*lat];
vel=data(1:5:NN,6); %velocidad(m/s)
rev=data(1:5:NN,5); %rpm
cm=(vel./rev)*1000;
N=size(vel,1)
x=[X'];
y=[Y'];
m=zeros(1,N);
z=[m,cm'];
t=data(1:5:NN,2);
%Initialize kml string:
kmlStr= '';

for i=1:1:N-1
    %Shift polygon by j degrees longitude:
   sx = [x(i),x(i+1),x(i+1),x(i)];
   sy=[y(i),y(i+1),y(i+1),y(i)];
   sz=[z(i),z(i+1),z(N+i+1),z(N+i)];
   for j=3:4
if 7.2<sz(j) & sz(j)<8.5
    sz(j)=7.9;
elseif 13.6<sz(j) & sz(j)<15
    sz(j)=14.4;
 elseif 19.6<sz(j) & sz(j)<22.2
    sz(j)=21;
 elseif 26.6<sz(j) & sz(j)<28.8
    sz(j)=27.7;
  elseif 33.4<sz(j) & sz(j)<35.5
    sz(j)=34.6;
else
    sz(j)=NaN;
end
```

Podemos ver que la relación no es completamente exacta, por lo que los que se encuentran dentro de un intervalo, les damos el valor exacto correspondiente, para evitar pequeñas oscilaciones. A continuación se muestran los resultados de correr el programa:



Los distintos tramos constantes son los distintos valores de la relación de transmisión, que evidentemente nos muestra en cada momento la marcha que está metida. Podemos observarlo mejor en la imagen siguiente ampliada:



Observamos cinco tramos distintos, uno inicial y claramente más corto que los otros que se debe a la primera marcha (color turquesa), y continuando la segunda como un tramo verde, la tercera en color marrón claro, en naranja la cuarta y finalmente la quinta en rojo.

El código completo del programa viene en el anexo con el nombre de “rectang_ctemarxa.m”.

5.5 - Adjuntar una leyenda

Como hemos podido comprobar las representaciones de los valores de las variables con la escala de colores son muy intuitivas para evaluar el mayor o menor valor de estas, pero no tenemos una leyenda que nos indique el valor de estas, aunque solo sea aproximado.

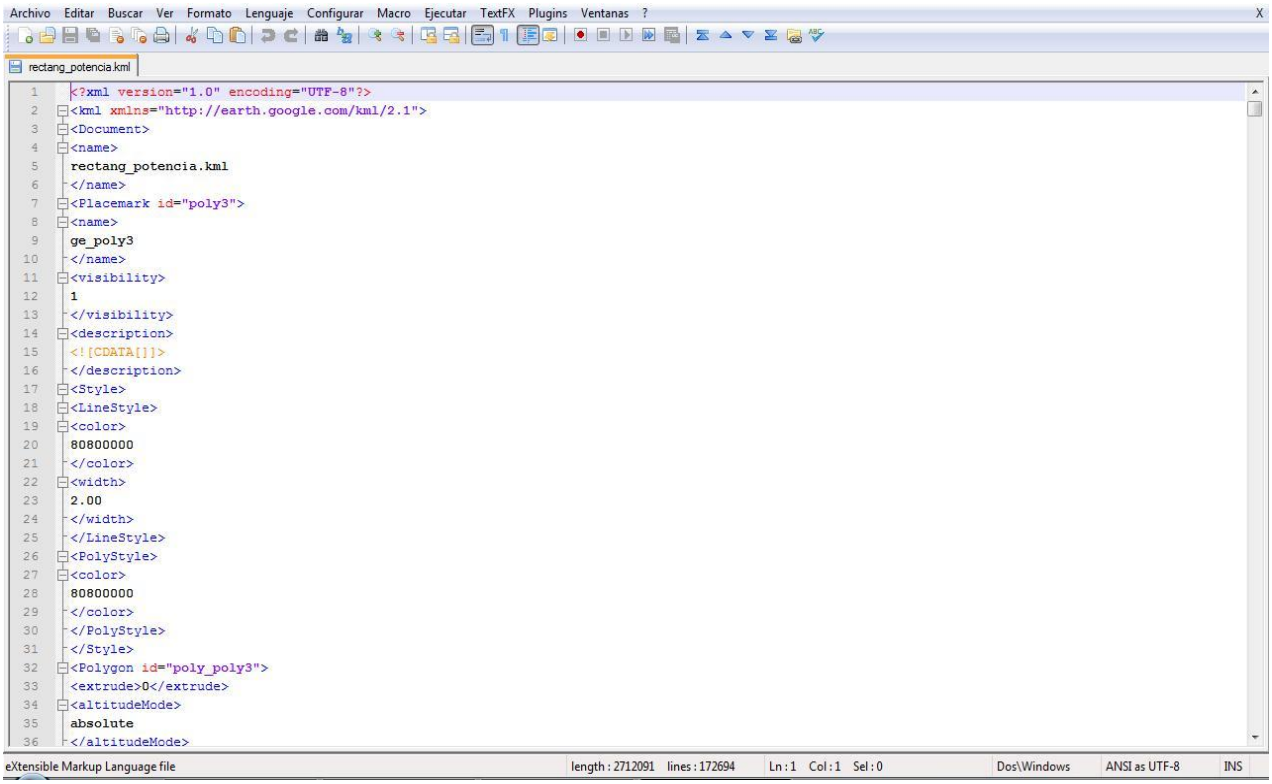
El sistema que hemos tomado para añadir leyendas en la pantalla de google earth es el de añadir una imagen superpuesta en la pantalla con la leyenda correspondiente.

Para hacerlo tenemos previamente que crear las imágenes correspondientes para cada una de las magnitudes que queremos representar. En el caso de la potencia, por ejemplo, debería ser algo como la imagen de la derecha, que no tiene por qué ser muy exacto, pero que de una buena idea de los valores:

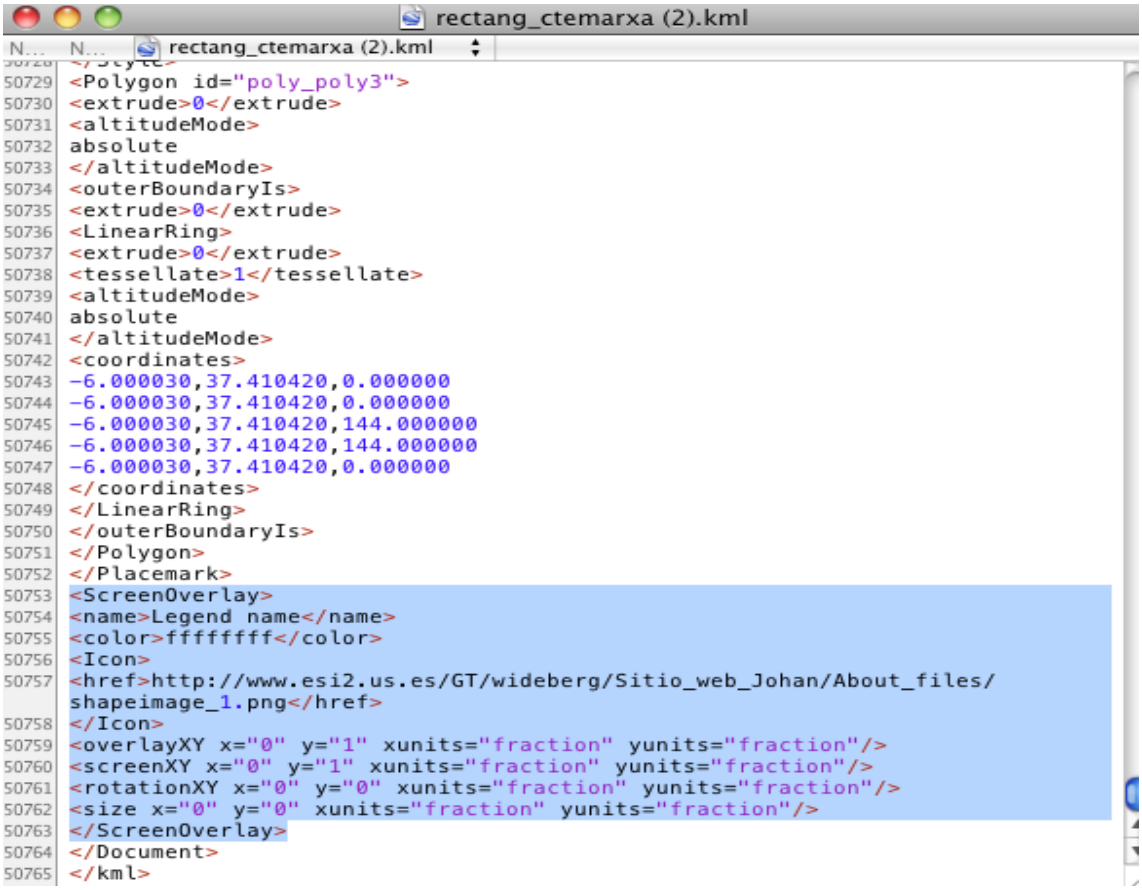
Una vez creada la imagen y guardada como archivo, se añaden unos comandos al archivo kml que se crea, que son los siguientes:

```
<ScreenOverlay>
<name>Legend name</name>
<color>ffffffff</color>
<Icon>
<href>http://server.com/logo.png</href>
</Icon>
<overlayXY x="0" y="1" xunits="fraction"
yunits="fraction"/>
<screenXY x="0" y="1" xunits="fraction" yunits="fraction"/>
<rotationXY x="0" y="0" xunits="fraction"
yunits="fraction"/>
<size x="0" y="0" xunits="fraction" yunits="fraction"/>
</ScreenOverlay>
```

Para introducir esto en el archivo de google earth, debemos abrir el archivo kml con un procesador de textos, en nuestro caso el Notepad ++, aunque puede ser cualquiera. Una vez abierto nos saldrá algo como lo de la imagen:



Entonces tendremos que introducir el trozo de código que hemos indicado anteriormente a una altura determinada. El lugar indicado para añadir la leyenda es el que indica la imagen:



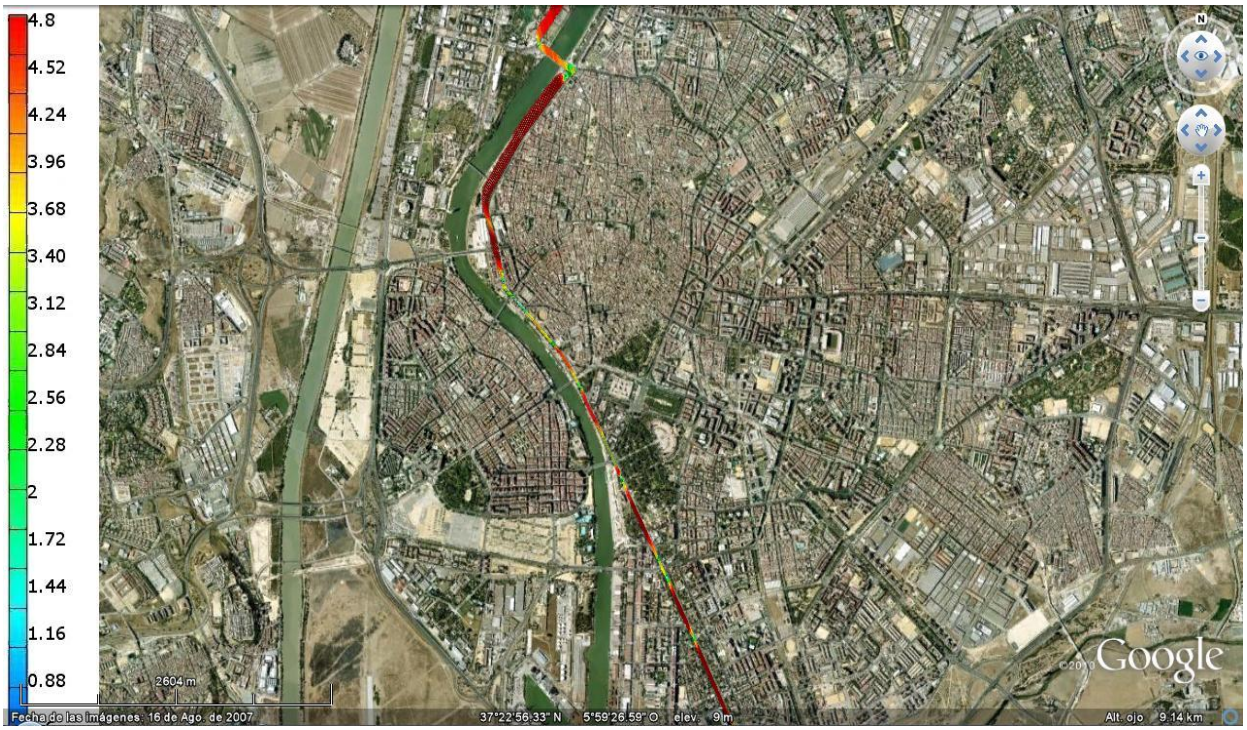
Una vez metido tenemos que hacer un cambio en el pequeño trozo de código que acabamos de introducir. Este cambio se debe a que tenemos que indicar donde se encuentra la imagen que queremos introducir. Esta localización debemos hacerla en internet, por lo que debemos de colgarla en una dirección web. Creando una dirección web donde situamos la imagen correspondiente a la leyenda ya podemos realizar el cambio. Este cambio a de hacerse en la línea siguiente:

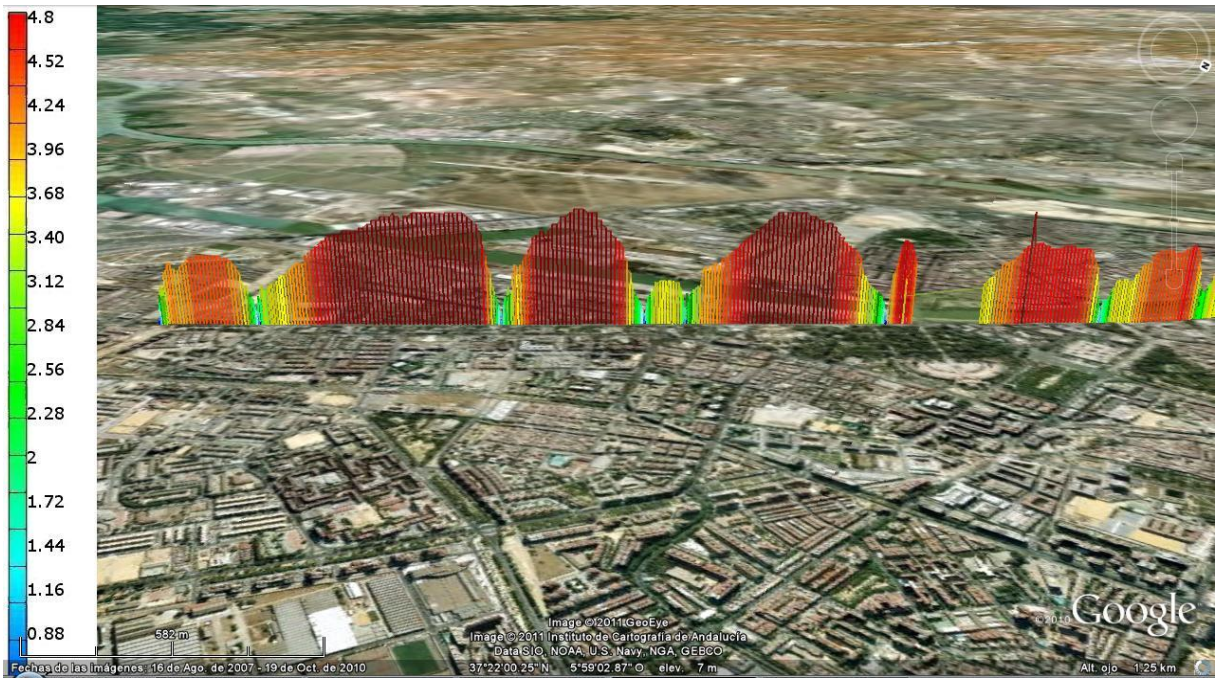
```
<href>http://server.com/logo.png</href>
```

Donde debemos de introducir, en lugar de la dirección web que observamos, la localización, colocando la dirección de internet correspondiente, del archivo de imagen que creamos antes. Una vez hecho esto lo que obtenemos es, para nuestro caso (donde nosotros hemos colgado la imagen):

```
<href>https://sites.google.com/site/proyectoobd/home/leyenda%20potencia.jpg</href>
```

Y si guardamos el archivo modificado y corremos el programa obtenemos finalmente la representación con la leyenda:




4.8
4.52
4.24
3.96
3.68
3.40
3.12
2.84
2.56
2.28
2
1.72
1.44
1.16
0.88
582 m
Image © 2011 GeoEye
Image © 2011 Instituto de Cartografía de Andalucía
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Google
Fechas de las imágenes: 16 de Ago. de 2007 - 19 de Oct. de 2010
37°22'00.25" N 5°59'02.87" O elev. 7 m
Alt. ojo 1.25 km