Geotagging Make your photos more useful with a little bit of coding

Geotagging: Tag

Marco Fioretti gets his holiday snaps out and adds more geographical information than you can shake a conspiracy theorist-shaped stick at.

Our expert

Marco Fioretti is the author of *The Family Guide to Digital Freedom* and is a free software activist and programmer.

Geotagging is the process of attaching to a generic digital object its geographical location. While geotagging generic data is a prerequisite if you want to analyse it with a geographical information system (see page 66 for more on this), geotagging digital photographs is a fun way to organise your photos by location, not just by date or theme.

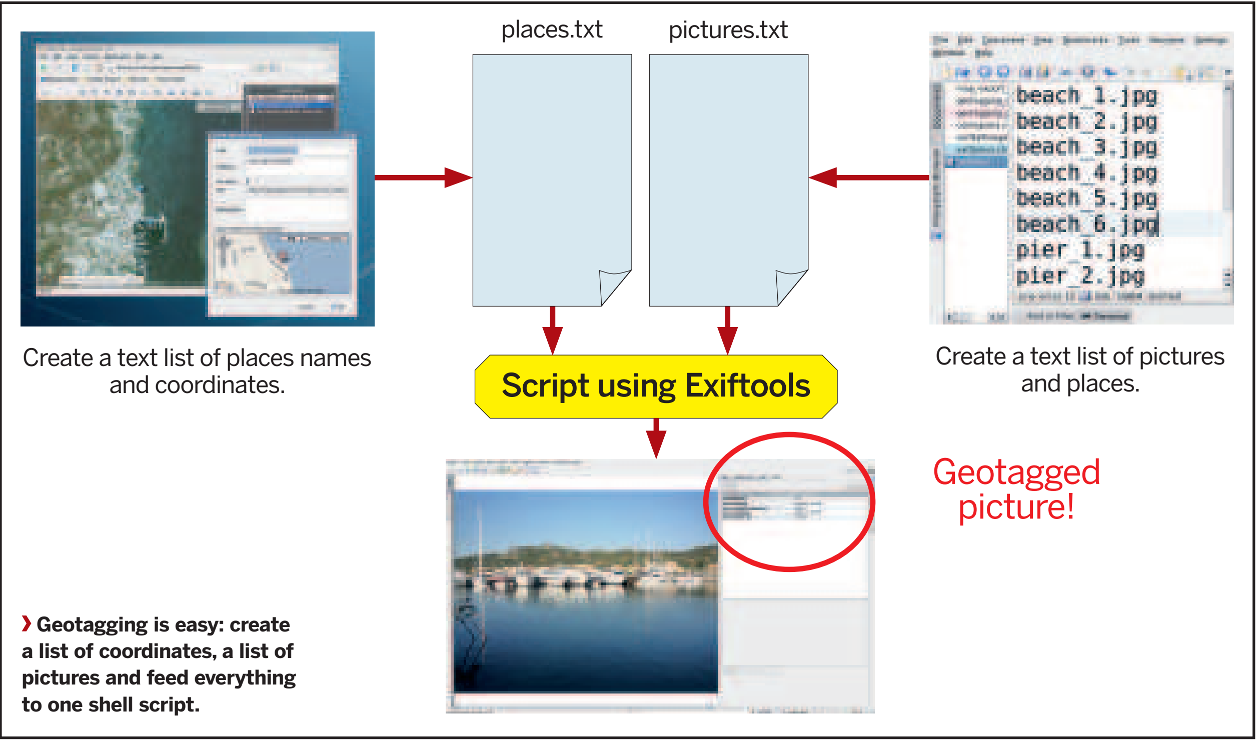
Geotagging photographs makes it possible to give your computer orders like "show me on a map where this picture was taken" or "find all my pictures taken within a three-mile radius of Buckingham Palace". If you want to publish your pictures online, geotagging makes it possible to make your own maps hyperlinked to and from your online picture galleries or services like Flickr.

Digital cameras with integrated GPS sensors that automatically geotag every shot will become more and more affordable over the next few months, but that doesn't mean that there's no reason to learn how to do it yourself. Think about it for a moment: if you like the possibilities of geotagged pictures, the biggest obstacle you're likely to face is not the shots you'll take from your next holiday, but the thousands of pictures you have on your hard disk already.

Keep it simple

What you're going to learn may not be the most sophisticated way to geotag but has all the characteristics to be immediately useful to all **LXF** readers who want to get started with geotagging because it doesn't require any GPS equipment. We're using free software that works without hassle on any Linux distribution around, and way it collects coordinates is so simple that you could delegate it to everybody who can type and handle a mouse. It also has the benefit of being faster than other GUIs, is easily extensible to add other types of taggings, and it covers all the geotagging needs of most non-professional photographers.

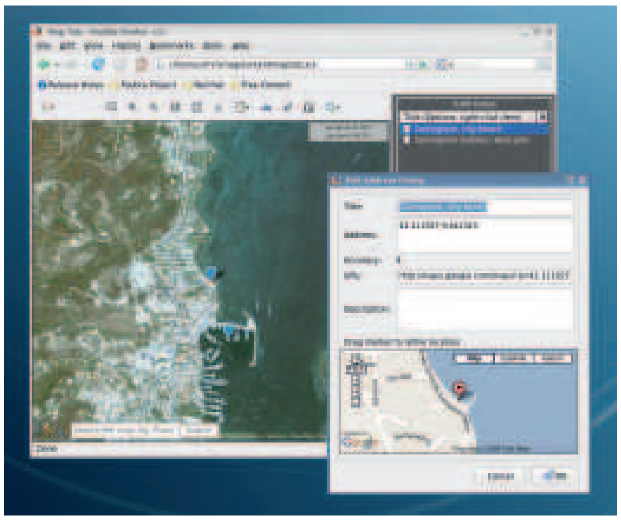
If you're ready, let's start looking how to geotag pictures with *Firefox*, its *Minimap* extension and just a little bit of *Bash* scripting,


places.txt
pictures.txt
Create a text list of places names and coordinates.
Script using Exiftools
Create a text list of pictures and places.
Geotagged picture!

```
beach_1.jpg
beach_2.jpg
beach_3.jpg
beach_4.jpg
beach_5.jpg
beach_6.jpg
pier_1.jpg
pier_2.jpg
```

› Geotagging is easy: create a list of coordinates, a list of pictures and feed everything to one shell script.

your photos



› **The *Minimap* extension for *Firefox* is probably the easiest way there is to create lists of geographic coordinates.**

using a few shots from wonderful Sardinia as source material. We're going to use *Firefox* to create one text file containing the coordinates of all the places shown in our photographs, and another plain text file that associates each shot to a place. This is all the information we need to get one simple *Bash* script to write the location of the shot inside each JPEG file, in a standard format that any geo-capable software can understand.

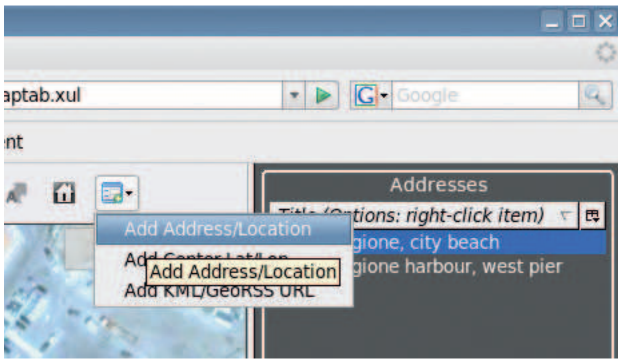
Trick question before continuing: which location should you geotag into a picture? The one of what you see in the picture, or the one from which the picture was shot? In this tutorial we use the second criterion, but there is no single right answer, really. You decide. And because we're data masochists, we'll note that in either case you should also geotag the orientation of the shot (looking north, looking south etc) if you really don't want to miss any geographical information.

Minimap

On to *Minimap* now: this is a wonderful *Firefox* extension that adds a side bar and a special tab for Google maps. The *Minimap* home page at **http://minimap.spatialviews.com** lists all the features of this plugin. Go read it, it's worth every byte. What we use here is just one capability of **Minimap**, shown above.

Pan and zoom as needed to select the exact point whose geographic coordinates you want to save. After that, left-click on the rightmost button in the *Minimap* toolbar, select 'Add Address/Location' and enter the name of the place and, optionally, a short description. When you click OK, that location will appear in the *Minimap* address list. If you right-click on it and open the Properties box, you'll see that *Minimap*, all by itself, has added the longitude and latitude of the place plus a Google Maps URL.

If you copy and paste that URL into any browser you'll get a Google Map of the same place with thumbnails of any picture of



› **This is where you click to save latitude and longitude of some place without even realizing it.**

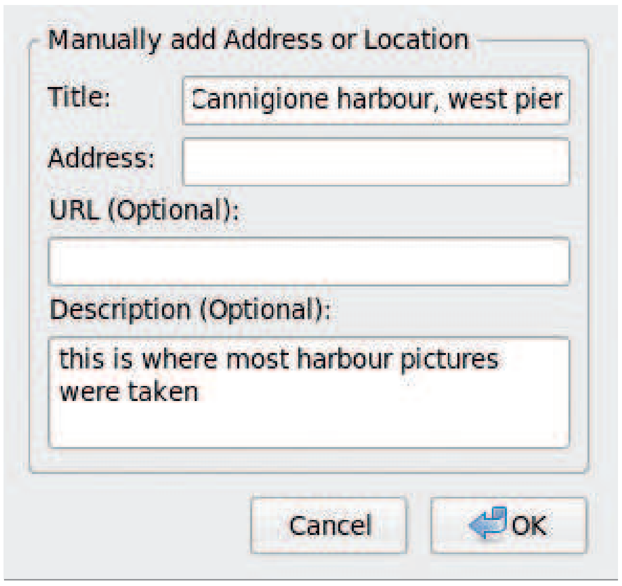
the same area available in the Panoramio database, but let's go back to geotagging. Repeat the procedures for all the places you need and, when you're finished, click on the leftmost icon in the toolbar and select Export All To > CSV. The result will be a file similar to this, which we called **cannigione.csv** (for clarity, only the first fields of each records are shown):

```
“Title”,”Location”,”URL”,”Description”,”Lat”,”Lng”
“Cannigione harbour, west pier”,”41.108790,9.442588”
“Cannigione, city beach”,”41.111927,9.441343”
```

Minimap can save addresses in other formats too, but we have chosen this because, as long as nobody uses double quotes inside descriptions or place names, it's very easy to parse:

```
marco@polaris ~]$ cut ‘-d”’ -f2,10,12 cannigione.csv | grep -v
^Title | cat -n > places.txt
marco@polaris ~]$ more places.txt
   1  Cannigione harbour, west pier”41.108790”9.442588
   2  Cannigione, city beach”41.111927”9.441343
```

»



Manually add Address or Location

Title: Cannigione harbour, west pier

Address:

URL (Optional):

Description (Optional): this is where most harbour pictures were taken

Cancel OK

› **You can add more details to the addresses you entered, or grab their Google Maps URL, at any moment.**

Quick tip

Even if you don't plan to use Google Earth right now, remember to save all your *Minimap* address lists in this format. You never know when they may become useful!

» With one short, albeit cryptic command, we have cut the CSV file in vertical slices delimited by double quotes, extracted the columns (2nd, 10th, 12th) which contain place name and coordinates, removed the unnecessary first line and numbered all the others. This is the list we'll use to associate latitude and longitude to pictures. Have you realised how simple it was to collect coordinates? Even better is that this part of the job only needs to be done once for each place on Earth and can be distributed among many people: everyone who's been on a trip to that destination can create partial lists, concatenate them in one CSV file, apply the trick above to that file only and share the results.

Even that auntie in Winnipeg who emailed you the pictures of the summer you spent there when you were 12 years old can help out. If she managed to send you an email, you can probably ask her to click a few more buttons to install *Minimap* and find all those great places whose names you can't even remember, even if she'll never know what geotagging or Linux mean.

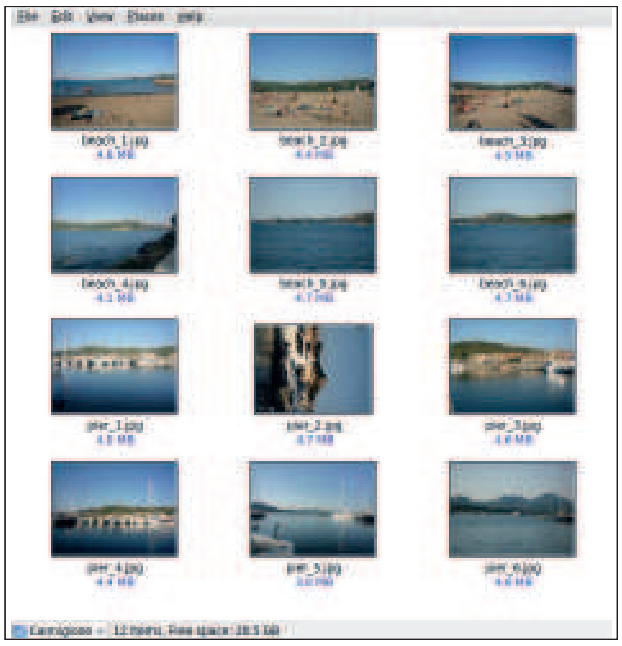
Come to sunny Sardinia!

The pictures we'll geotag were taken from the pier and the beach. We are going to write the latitude and longitude inside each file in the Exchangeable Image File Format (Exif, **www.exif.org**), the *de facto* standard for embedding many kinds of metadata inside JPEG pictures. All digital cameras write lots of Exif data in their shots. A common way to read and write this data under Linux is with the *Image-ExifTool* Perl modules (**http://cpan.uwinnipeg.ca/dist/Image-ExifTool**) and their command line interface, called *exiftool*. The installation of this software is very simple: download the zipped tar file, expand it, enter the source code directory from a terminal window and issue (as root) the following commands:

```
perl MakeFile.PL
make
make test
make install
```

(you may need to perfom the same procedure for other Perl modules necessary for *Exiftool* to work, if they aren't already installed on your computer).

This is a very short excerpt, made with *Exiftool*, of the Exif tags hidden inside a generic digital picture:



› Geotagging is also a fun and easy way to learn geography: here's Cannigione, north-west Sardinia.

```
[marco@polaris Cannigione]$ exiftool pier_1.jpg
 ExifTool Version Number        : 7.30
 File Name                      : pier_1.jpg
 Exposure Program               : Landscape
 ISO                            : 160
 Date/Time Original             : 2008:08:07 07:52:08
 Title                          :
```

What we need to do in order to geotag our JPEG files is to write inside them, in Exif format, latitude, longitude and elevation. The GPS part of Exif is described at **www.sno.phy.queensu.ca/~phil/exiftool/TagNames/GPS.html**. This page shows that there are many GPS tags, but the only mandatory ones are: GPSLatitude, GPSLongitude, GPSAltitude, GPSLatitudeRef, GPSLongitudeRef, and GPSAltitudeRef. The last three are the references for the respective fields, whose possible values are listed in this table:

GPSLatitudeRef:	'N' or 'S' (north or south of the equator)
GPSLongitudeRef:	'E' or 'W' (east or west of greenwich)
GPSAltitudeRef:	0 or 1 (above or below sea level)

In our case, we'll set **GPSLatitudeRef** equal to 'N', **GPSLongitudeRef** to 'W' and **GPSAltitudeRef** to 0. Latitude and longitude for each location are already available in the **places.txt** file. The only thing we're missing is the altitude. The pictures of this example are all taken at sea level, so we can set it to 0 and go on, but the Google Maps API used in *Minimap* always returns 0 for altitudes, so until that changes, *Minimap* will continue to ignore this field. On the other hand, such a limit only matters if you really need to add the exact elevation of every place you photographed, that is only if you think you'll need to find, some day, all your pictures taken between 1,000 and 1,200 metres above sea level, or something like that.

Oh, yeah, geotagging: relax, we're almost there. The list of JPEG files to geotag is a plain text file, here called **pictures.txt**, in a very simple format: one filename per line, separated by a colon from its location number, as found in the first column of the file **places.txt**.

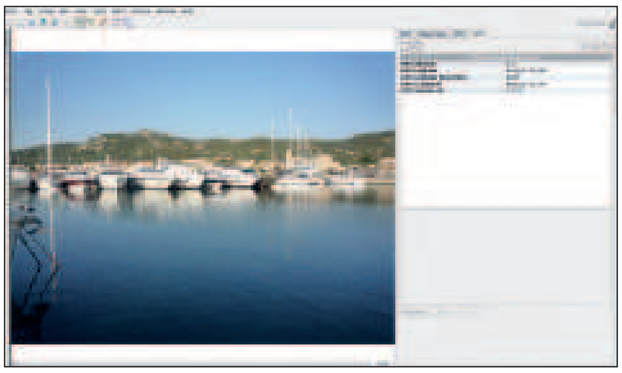
```
beach_1.jpg:2
...
pier_.jpg:1
```

Now that we have everything in place, let's move to discuss the last piece of the flow diagram on page 92 – the shell script called **geotagger.sh** which, in about 20 lines of code, does the real work. Here is a listing of the core logic:

```
##################################################
while read line
 do
   PLACE_NUM=$(echo "$line" |awk '{print $1}')
   LAT=$(echo "$line" | cut -d\" -f2)
   LONG=$(echo "$line" | cut -d\" -f3)
   LONG_ARY[$PLACE_NUM]=$LONG
   LAT_ARY[$PLACE_NUM]=$LAT
done < $1

while read line
 do
   PLACE_NUM=$(echo "$line" | cut -d: -f2)
   CURRENT_FILE=$(echo "$line" | cut -d: -f1)
   echo Associating LAT: ${LAT_ARY[$PLACE_NUM]}, LONG:
${LONG_ARY[$PLACE_NUM]} to $CURRENT_FILE
   exiftool -P -c "%.6f degrees"              \
        -GPSLatitude=${LAT_ARY[$PLACE_NUM]}   \
        -GPSLongitude=${LONG_ARY[$PLACE_NUM]} \
        -GPSAltitude=0                        \
```

» **Never miss another issue** Subscribe to the #1 source for Linux on p102.



› **Once a JPEG is geotagged according to the Exif standard, any digital picture manager can recognise and use that information.**

```
        -GPSLatitudeRef=N               \
        -GPSLongitudeRef:W              \
        -GPSAltitudeRef:0               \
        $CURRENT_FILE
  done < $2
```

The script needs two filenames as arguments. The first one (**$1** in the listing above) is the list of place names and coordinates; the second (**$2**) contains the picture names and their location numbers. Everything happens in two 'while' loops, both of which scan one file, one line at a time.

To begin with, we load the longitude and latitude of each place into the two arrays called **LONG_ARY** and **LAT_ARY**, indexing them by place number. The second loop reads the list of pictures with the place number associated to each of them. The script then calls the *Exiftool* utility with the right parameters. That's it.

Exiftool has a very detailed man page, which you should really read before you try this at home, but its usage in this script is pretty self-explanatory. The first two options preserve date and modification time of the JPEG file and set the print format for the GPS coordinates, while the others simply assign the right value to each parameter. The script must be run in the directory where all the pictures are:

```
  [marco@polaris Cannigione]$ geotagger.sh ~/places.txt ~/
 pictures.txt
  Associating LAT: 41.111927, LONG: 9.441343 to beach_1.jpg
  ...
  Associating LAT: 41.10879, LONG: 9.442588 to pier_6.jpg
     1 image files updated
```

after the script is finished you'll find both the geotagged pictures and a backup copy of the original files labelled with the **_original** extension. Repeating the same test made at the beginning of the tutorial will show that the GPS Exif tags were indeed added:

```
  [marco@polaris Cannigione]$ exiftool pier_1.jpg | grep GPS
  GPS Version ID      : 2.2.0.0
  GPS Latitude Ref    : North
  GPS Longitude       : 9 deg 26' 33.32"
  GPS Altitude        : 0 m
  GPS Latitude        : 41 deg 6' 31.64" N
  GPS Position        : 41 deg 6' 31.64" N, 9 deg 26' 33.32"
```

Smart tagging

As the Alternative Solutions box above-right demonstrates, this method isn't the only one around. Besides, it doesn't guarantee the maximum possible precision: if you need military-grade resolution of geographical coordinates, this isn't the right solution, sorry. The method explained here, however, has lots of advantages. All you need is one *Firefox* extension, one Perl module and one shell script of 20/30 lines. The major advantage of this method, however, is how it partitions the two phases of the job, namely coordinate gathering and JPEG file tagging. The real obstacle when somebody decides to geotag 10+ years worth of family pictures is entering the coordinates of all the places. Working as explained here, the coordinates collection phase is extremely simple, completely separate from the actual geotagging procedure and works on every operating system. Even more important, different people can perform it in parallel, independently, one bit at a time. The geotagging script, on the other hand, is simple, fast, completely automatic and in many real-world cases can be simplified even further. If you have some hundred pictures all taken in the same place, for example your home, you could just remove the first 'while' loop and hard-code home latitude and longitude in the second one!

Another important characteristic of this flow is how easy it is to automatically extend it beyond geotagging. There is nothing which prevents you from adding to the **pictures.txt** file columns like "Gallery name", "People visible in this picture", a scene description and so on. If you do that, *Exiftool* can write all those extra metadata in the same way as it does with geographic coordinates, just check out its man page to know how.

Here's a final treat, shown below. *Minimap* can save address lists also in the KML file format used to bookmark and annotate places with Google Earth. With just a bit more scripting, you can use that KML file and the same lists you create manually or through *Minimap* for geotagging to create albums that can be browsed with Google Earth! In the meantime, start geotagging! LXF

Alternative solutions

The 0.10.x releases of *DigiKam*, which will probably be available in most Linux distributions from the beginning of 2009, will be able to load KDE 4's *Marble* map widget (**http://edu.kde.org/marble**) and use it in a manner similar to *Minimap*. The blog entry at **http://linuxappfinder.com/blog/add_gps_coordinates_to_your_photos** explains how to do the same thing manually, with *DigiKam* under KDE 3.x.

You can also do GUI-only geotagging with *Geotag* (**http://geotag.sourceforge.net**). This Java program is also capable of launching Google Earth or Google Maps to show where a picture was taken, and can generate KML files that other Google Earth users may load to do the same. You can also ask it to find more detailed information about a place by looking it up on the **geonames.org** website and, if the right version of Java is installed, run *Geotag* from *Firefox* without installing it (see **http://geotag.sourceforge.net/?q=node/3**).

Finally, if you feel like coding a bit more than we did in this article, you can download the Perl modules for GPX and Flickr programming from **www.cpan.org**.

› **All you learned in this tutorial can be easily extended to distribute your picture galleries in a format usable with Google Earth.**