Google Maps And Image Mashup Final Report

Trond Stokkeland

December 2, 2009

Contents

Planning **2**
- The Problem . . . 2
- The Goal . . . 2
- Means . . . 3
- Methods . . . 3
- Milestones . . . 3
- Critical Elements . . . 4
- Resources . . . 4
- GANTT . . . 5

Implementing **6**
- Collecting data . . . 6
- Describe . . . 6
- Transform . . . 7
- Move . . . 8
- Presenting using Mashup (KML) . . . 9
- Automation . . . 10

Documentation **11**
- End product . . . 11
- Conclusion . . . 11
  - Process . . . 11
  - Product . . . 12
  - Suggested future work . . . 12

Planning

The Problem

Being able to match images taken off the same building but 150 years apart can be quite a bit of work. First you must have a 150 year old image and make a digital copy off it. Then you have to find out where the picture was taken and take a new picture from the same place in the same angle. Then create a mashup with Google Maps in order to get it placed on a map. All of this have to be done for each set of images you have.

If we assume you have found a old picture to start with and you have good knowledge about the local area, finding out where it was taken shouldn't take too much time, let's say ten minutes per picture. We can be nice and say that during those ten minutes you also take the new picture. Thats all good. But when you come home you need to match all the pictures, if you took them in a specific order you might do it in a few minutes, lets say two minutes. If you then have spent about a day to create a script that creates KML files for a set you still need to run it once per image. If you then need to spend maybe two more mintues on editing the script for each set of images. You will then have spendt fourteen minutes per set. With one hundred original images that will take almost twentyfour hours.

The Goal

Today it is fully possible to get cameras that take pictures with geo-data in their metadata. The process of matching images is something that could be automated but it would require a alot in form of image recognition. Another project that should have been made is a easy way of browsing two sets of images, then if you find a match you can with a button export geo-data from one image and insert them into the other.

This project is all about what happens after you have matched images and they all have geo-data in their metadata. The idea is that all you have to do is upload the images to the server and the rest will happen there. That should

make it possible to save maybe two minutes per set, with one hundred images that should save you almost three and a half hours work.

Means

The job will be done using a bash scrip that extracts the meta data from the images using exiftool and compares the information. When it finds a match it generates a KML file linking the matching images with proper placement on Google Maps. To automate the process abit more it is just to place the script in the servers scheduler and run it every so often.
To prevent the script from going throught the same files over again they can be moved to a different folder. That way the script wont take up much CPU time when there are no new images.

The legal implications in this project is the use of historic city photogrphs from the museum collection. The product of this project will be released under the [2]creative commons BY-SA-NC lisence. The product include the code, the generated KML files and pictures taken.

Methods

Any text editor will do to write the bash script in.
Exiftool will be used to extract the metadata.
Imagemagick will be used to resize the images.

Milestones

1. Finish pre-project report
2. Create test images with Geo-taggs
3. Photograps aquired from the museum.
4. Photographs taken
5. Script matching images
6. Script moving matched images to seperate folder
7. Script generating KML for a set of images
8. Script open a exsisting KML and adds a new set of images
9. Write final project report

This list is created as reference for the milestone column in the GANTT diagram.

Critical Elements

- Script matching images
- Script generating KML

Resources

- Borrow camera from Høgskolen in Gjøvik.
- Get old images and location from the museum.

GANTT

Milestone	Activities	Resources	Dependencies	Risk	45	46	47	48	49	Hours
Pre-Project										
1	Writing the pre-project report				15					15
Final Project										
2	Take two random images and insert geo-taggs	Two images				1				1
3	Get photoes from museum	Museum		Data corruption		1				1
4	Take photoes	Camera	Borrow camera	Data corruption		2				2
5	Code matching part of script		Test images or real images	Bugs in code		2				2
6	Code file moving		files	Bugs in code		1				1
7	Code KML generation		Image matching	Bugs in code			5			5
8	Code KML editing		KML generation	Bugs in code			2			2
9	Write final project report		Done with everything					7	8	15
SUM					15	7	7	7	8	44

Implementing

Collecting data

The project began with finding 13 images from Mjøsmuseet, [6]Gjøvik's website. All of them was taken around 100 years ago, and didn't realy have much metadata at all. Then in week 47 Christian Hochlin and I went to Mjøsmuseet, Gjøvik to get locations of the images marked on a map, this way we knew approximatly where the pictures were from. The same week Christian Hochlin, Pelle Bjerkestrand and I went out to take pictures off the same locations. The problem with taking pictures 100 years later is that alot of things change, trees grow, new buildings stand in the way and old buildings have fallen down. But all in all it went pretty good. When this was done we had eleven usefull sets of images.

When taking the new pictures we used a [4]Nikon D3000. With the camera came a [5]GP-1 GPS receiver that refused to work with the camera. What we ended up doing was to take the same pictures with a iPhone. This way we had a source for the correct gps metadata.

Describe

As my project is about sorting and generating kml for well described images, I had to make sure the images were well described. In order to make them well described I had to insert a few things into the metadata of every image. In the images taken by Christian Hocklin, Pelle Bjerkestrand and me I inserted autor,copyright information and location name (ImageDescription) before I started to look at the GPS data.

As neither our old or our new images had any GPS data in them I had to extract the GPSLatitude and GPSLongitude from the images taken with the iPhone and insert it into the old and new images. A problem I had with the metadata here were that it is a field called GPSPosition that consists of both GPSLatitude and GPSLongitude, but of unknow reasons I wasn't allowed to use that field. So the script now have to check both GPSLatitude and GPS-

Longitude instead of only GPSPosition. This is very bad when it comes to the performance but atleast it works.

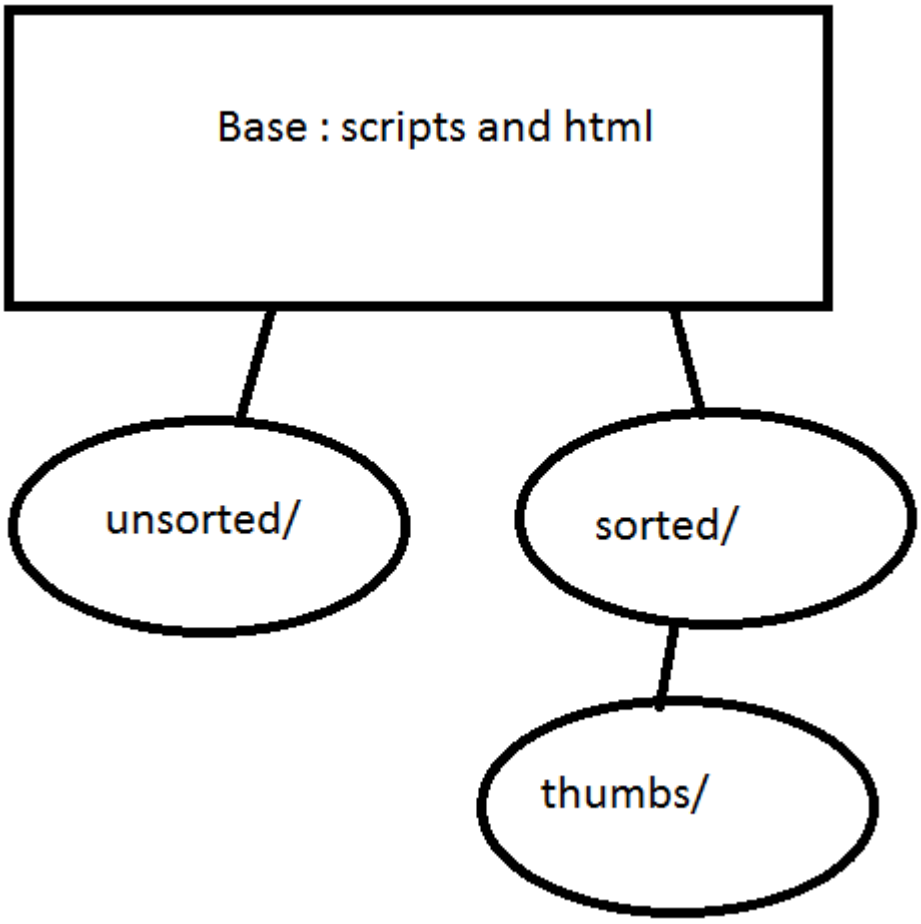
Another minor problem with the metadata was that the old images didn't gave any creation date inn them. So when the script find a image without a date tag it sets the date as unknown

Transform

Using [8]ImageMagick's convert program to create thumbnails for every image to reduce the size of images and the amount of data needed to load. The thumbnails link to the original images so that those interested still can access the original images. It can be discussed wheather the linked images also should be lowered abit in size, but for this project I dont see the need.

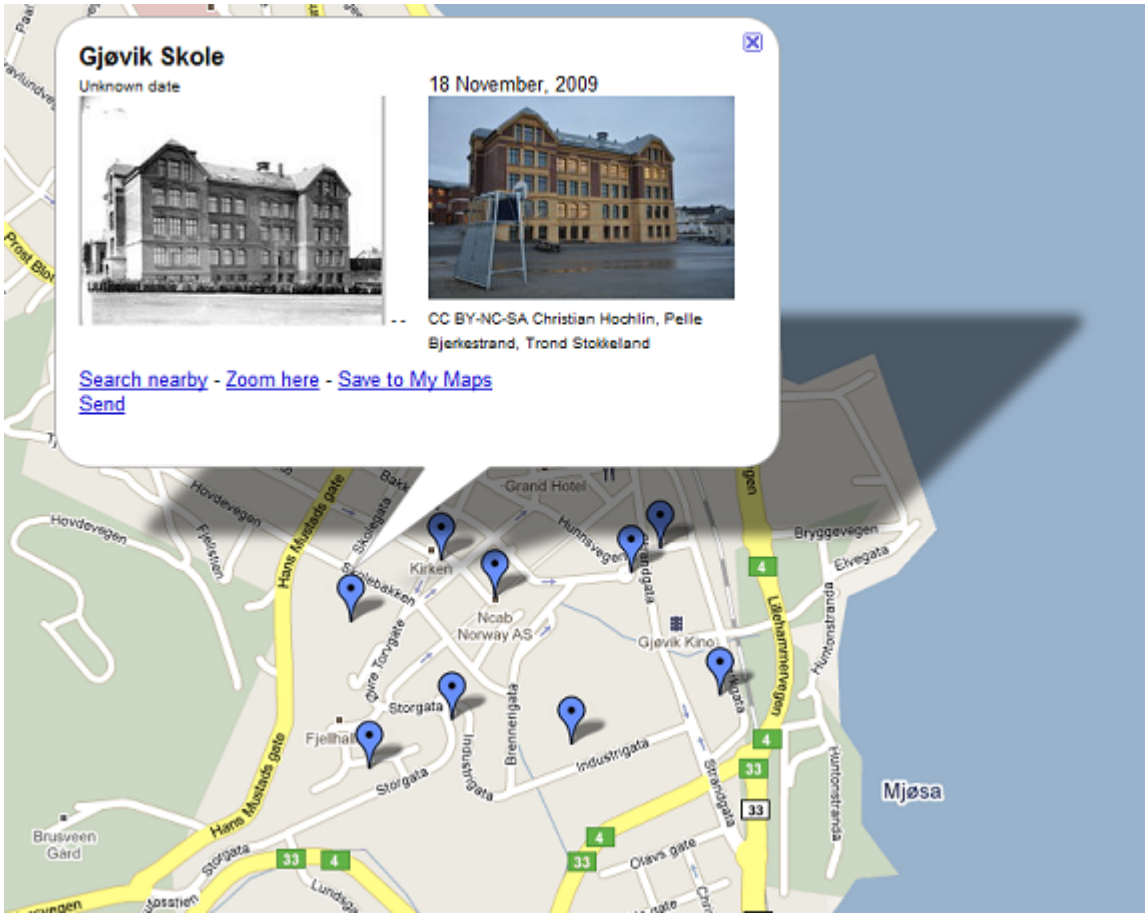
To give some numbers. The original images from the museum are from around 40 to 60 KB, while the thumbnials are down to 4 to 13 KB. The space saving is pretty good thinking percentage here and it gets even better for the new images. While the originals vary form 2.8 to 3.3 MB the thumbnails are down to around 60 KB.

Move



I found the easiest way to arrange the images is to have one folder for the unsorted images one for the sorted and a subfolder for the thumbnails. This makes it to figure out when there are new images to sort, simply by checking if the unsorted folder is empty.

Presenting using Mashup (KML)



The presentation of the data is created by using a mashup of old images from the museeum, new images from my work and it all placed into a KML document that can be inserted into google maps as showed in the image. I understand that it is easy to create a kmz file from the kml file, it is just to pack it into a zip archive and then renaming it to a .kmz file. There were two reasons for me not doing this. The first is that the project suggestion said to generate a kml file, and the second reason is that in the [1]example in suggestion one there is also used a kml file.

Automation

The automation process here is easy. Just to put new images into the unsorted folder, and whenever the server scheduler makes a call for the sort.sh script the script will check for files If it find new files it will create a kml file, or add the images to a excisting kml file. When that is done it will move the images to a folder for sorted images.

The idea here is that when installing the script on a server you just place the script files and the folders where you want them. After that you only have to put the main script into the servers scheduler to make it run automatic updates. What I have heard is that adding the script to a the [9]Crontab file would be the way to do this. I have not set up the contab on the loke server. This is partly because Im not sure I have access to do that and partly because of problems with ssh unables me to try for the moment.

Documentation

End product

Images taken by Christian Hochlin, Pelle Bjerkestrand and me are used side by side with the images from the museeum

generate.sh is the script that takes two images as arguments creates thumbnails and places them into the kml file, for the first run it generates the kml file.

The sort.sh the main script that sorts the images and calles on generate.sh when it finds a match before it moves the images to the sorted folder.

The gjovik.kml is the kml file generated by generate.sh .

echo (sed'd' KML) >KML

For an unknown reason the part of the script that removes the kml foot tag also removes the formatting of the file. So Everything except for the last placemark is hard to read in the file.

[7]The website shows abit about the project and how the result ended up. All the images and the script files are accessable via the page. This is with exception of the images taken with the iPhone, because I did not see any reason to put them up on the webpage.

Conclusion

Process

The gathering data part took abit more time than expected. Getting information from the museeum went as planned, but taking the new images took us around three hours. What took so much time was walking around to the different places and making sure we were at the right spot.

The coding also took more time than expected, instead of the total of ten hours I've spend about 15 hours. This is becuse this is the first bash script I ever write that is more than a couple of lines. So I used alot of time looking up how things work, and what commands could be used.

The webpage was not a part of the plan because it is only a simple webpage used to present the work. Still it took me about two hours getting everything on the page as it is now.

Product

The end product does what I had in mind when starting this project. It makes it possible to upload images to a folder on the server, then the script will match the images and add them to the kml file.

Suggested future work

Features that could be added to the script:

- Relative paths, now the script requires the images to be in a specific folder. Would be nice to pass the path as an argument to the script.
- Filenaming, now the script assumes that no images has the same name.
- Errorchecking, if something isn't as its supposed to be the script wont take that into account. Eksample if metadata is missing or if the filemoving fail.
- Multiple images from one location. Might be handy to support several images taken at the same place.

Bibliography

[1] , *Refsvik, Kjell Are 20091021¡mt4951¡ntroduction.pdf*

[2] , *creative commons BY-SA-NC lisence creative commons BY-SA-NC lisence*

[3] , *Cooper, Mendel Advanced Bash-Scripting Guide*

[4] , *Nikon D3000 Camera information*

[5] , *GP-1 GPS receiver GPS information*

[6] , *Mj{*osmuseet http://www.mjosmuseet.no/

[7] , *The website Website*

[8] , *ImageMagick http://www.imagemagick.org/*

[9] , *Crontab Crontab - Quick reference*