STUDIENPROJEKT GEONETCAST WS 2009/10

Automated Selection and Processing (ASAP)

Technical Documentation

Harald Hecking, Martin Kurowski

12.02.2010

Index

Chapter 1: Selection process

The selection process is done in several steps. At first, ASAP loads a given set of n images. Then, it checks for KO-criteria: If a picture doesn't have enough contrast or is of bad quality, it is sorted out and not regarded anymore.

Next, the images have to be checked spatially. Of course it only makes sense to compare those images, which have the same spatial extract. How can you tell, if a picture is similar to another, if it shows a different space? The images are geotiffs and therefore they store spatial information from which you can conclude the bounding box. By comparing the bounding boxes, three cases can occure: Firstly, both images match spatially, hence they belong to the same set. Secondly, they can partly overlap. If the percentual overlap is beyond a certain threshold, the images belong to the same set. From now on, only the overlapping part is regarded to measure similarity. Thirdly, both images are spatially different. Now, they cannot be compared to each other, but maybe with other images of the input. When all images have been spatially checked the result is a set of sets.

In the next step, each of the sets is checked for similarity: For each combination of two images in the set, a mahalanobis distance is calculated. The higher the distance is, the more unsimilar the images are. The mahalanobis distance is used to take into account the spatial structures of the images. For the statistical details, see chapter 4.

When all distances are calculated, the most representative images can be chosen and finally be stored on the server.

Chapter 2: Architecture

ASAP is created in a classical three tier structure (see image 3.1) and programmed in the object oriented language JAVA. First, we have the GUI. Since the GUI is not our main focus of interest it is held quite rudimentary. You can load the images you want to check and secondly define the search parameters, like the quota which defines the percentage of how many of the images you want to select. The GUI transfers the selection parameters to the manager. Next, the manager calls the file loader. The file loader loads the images from the GNC server into the programm. In order to avoid a heap space overflow, we don't use the actual images with all its huge amount of data store. The file loader creates a proxy object. This is a classical software design pattern. It contains only the relevant information of the image. That is its bounding box, its file

adresse and a sample of the relevant image pixels. We don’t analyze all of the pixels since the statistical computations that will be performed afterwards are very complicated. Thus, we can only take a random sample of information into consideration. A sample is created randomly, but for each image the same pixels are chosen. A quite exact but still performant number of pixels is 1000. Each sample consists of sample element. This is the representation of a pixel. It contains the image coordinates and a set of values for each channel in the image’s spectrum. When we created a sample for each image in the GNC-input folder, a set of proxies is created and returned to the image manager to be analysed afterwards.

Next, the image manager transfers the images to the spatial checker. The spatial checker serves to select the images spatially. The idea is to compare those images which have the same spatial extract. It would not make sense to compare two images concerning similarity if they show a different space. Therefore it checks the images and sorts them by their space they represent. This results in a set of sets of images of the same spatial extract.

Each of those sets is next analyzed statistically in the similarity selector. For the complex statistical calculations we use an java based interface to the statistical software R (rjava). At first, each image is compared with each other image. The mahalanobis distance is a measurement of similarity. It is small if two images are similar and vice versa. All distances are represented in a distance matrix. The distance matrix is analysed by a one dimensional scaling. This brings the images into an order. Now the most representative images can be selected. The most representative ones are sent back to the fileloader. Concluded from the adresse stored in the proxy object, the fileloader copies the most representative images into the target folder. How the statistical mechanism really work will be described in detail in the next chapter.

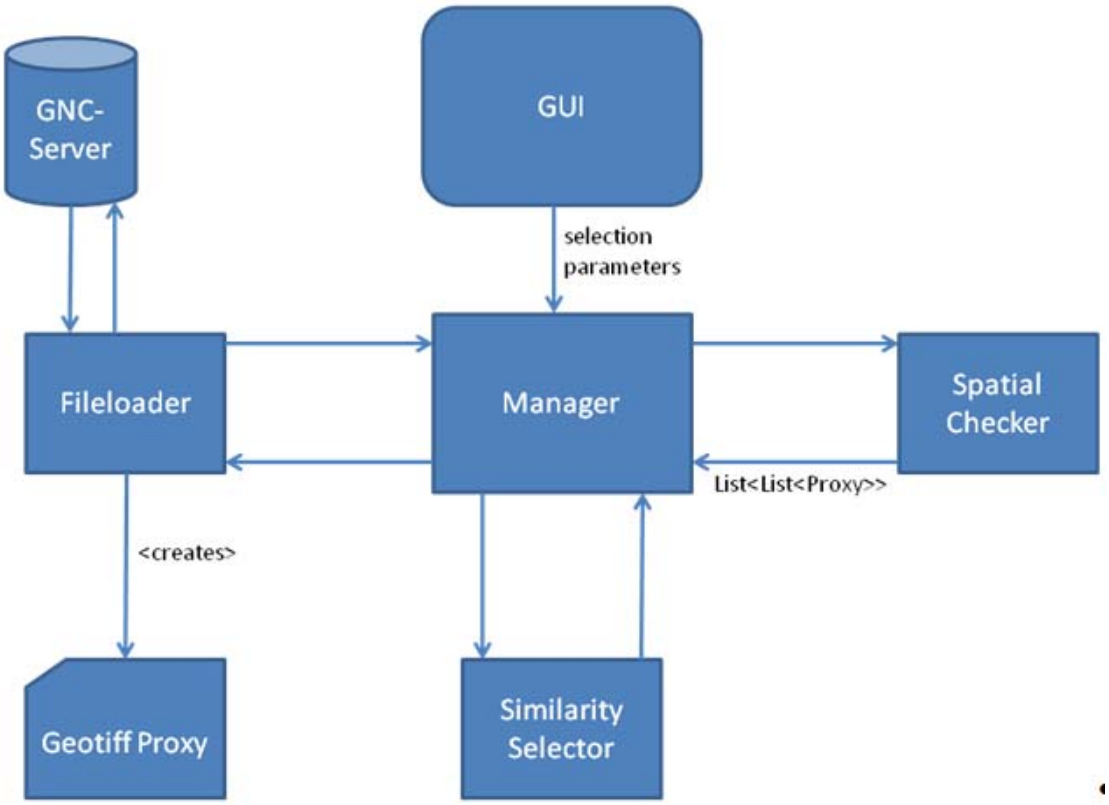

Figure 1: ASAP Architecture.

Chapter 3: Statistical Methods

This chapter describes the statistical methods used in the class „Similarity Selector“. To select the representative images of a set of images, it is necessary to define a measurement of similarity. The mahalanobis distance is a way to calculate that (chapter 4.1). To calculate the mahalanobis distance it is first important to estimate the autocovariance function of an image to use the spatial information contained in the image. (chapter 4.2). When all of the images have been compared to each other, we get a distance matrix of all combinations of images. To get the representative ones, we have to project the similarity information on a scale. The technique used to do that is called multidimensional scaling.

3.1. Mahalanobis Distance

The Mahalanobis Distance is a measurement for similarity between two images x and y and is defined as follows:

$$d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y}) * S^{-1} * (\vec{x}-\vec{y})}$$

$$\vec{x} = (x_1, x_2, x_3, \dots, x_n)$$

$$\vec{y} = (y_1, y_2, y_3, \dots, y_n)$$

$\vec{x}$ and $\vec{y}$ are representing the samples that are taken out of the images. Each x_i is an element of the sample and represents a pixel value. In case it is a multi spectral image, it is a vector of pixel values, one for each channel. The samples are taken for all of the k images that have to been checked for similarity. The sample size is n and the sample elements are taken for each image at the same position. We cannot check every pixel of the image since the computational effort would be way to high. That's why we take a sample of a thousand pixels into concern.

As we see, the formula of the Mahalanobis Distance looks quite similar to the euclidean distance. But instead of just taking the pixel-wise distances by square, those distances are weighted by the inverse matrix of S. What is S?

$$S = \begin{pmatrix} cov(X_1, X_1) & cov(X_1, X_2) & \cdots & cov(X_1, X_n) \\ cov(X_2, X_1) & cov(X_2 X_2) & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ cov(X_n, X_1) & \cdots & \cdots & cov(X_n, X_n) \end{pmatrix}$$

S is the covariance matrix containing the covariances of all pixel combinations. It is inverted and then used to weigh the pixels. The idea is to take into account the spatial information of the image. Lets say we want to select the best images out of a set of 50 images by checking 1000 pixels. Since k, the number of images is way smaller than n, the number of sample elements, the covariance matrix would be singular and therefore not invertible. The next sub-chapter shows, what can be done.

3.2. Estimating the auto-correlation function

Instead of calculating each covariance, we assume that the covariance $cov(X_i, X_j)$ only depends on the distance between the two pixels to be compared (X_i and X_j). The distance is the euclidean distance between the image coordinates. At first, we need a standardized function that depends on the distance. We use the following function: $acv(d_{i,j}) = e^{-d/a}$

The name acv stands for autocorrelation function. It is estimated by a non-linear regression between the distance and the correlation coefficient of X_i and X_j. If the distance is zero (ergo it is the same pixel) the function has the value 1 which makes sense because the correlation is trivially 1. If the distance gets larger, the function value

converges to 0 which makes sense as wells, since the spatial influence diminishes with growing distances.

Now, we have a function that results in correlation values depending only on the distance. The concept is similar to stationarity in time series analysis.

From the correlation values we can easily calculate the covariances as follows:

Next, the matrix S can be estimated and inverted. The distance matrix of all combinations of images can be calculated by computing the Mahalanobis distance for each pair of images. Now that we have a similarity measurement for images we have to decide on which images are the most representative ones.

3.3. Multidimensional scaling

To find the most representative images we have to create an order of images. For that, we use a statistical method called multi-dimensional scaling. To describe what it is in detail, let‘s show it by the following example: Let’s assume, we have a distance matrix of 5 locations. From that we want to conclude their relative positions. With a two dimensional scaling we can find a two dimensional order of those five locations (ergo their positions in a 2D-space). For our purposes we use the same method, but instead of two dimensions, we want to have a one dimensional order. Precisely, we use „one-dimensional scaling“. As input we take the distance matrix, as output each image gets an MDS value. The set of images is ordered by the MDS value. Next, the most representative images are those who are the percentiles of the ordered set of images. Those images are selected and the result of our statistical evaluation.

Im10	im1	im7	im2	im6	im8	im4	im3	im9	im5
-500	-312	-201	-23	65	134	333	506	709	899

Figure 2: Example of One Dimensional Scaling.

Chapter 4: Test and assessment of the results

First we tested ASAP for its stability on the Operation Systems Windows XP, Windows Vista and for UNIX systems Mac OS. After the stability tests we came to the conclusion that ASAP is running in these enviorments. We had not the capacity to test different hardware settings but we suppose it runs within the most common hardware settings.

Inconsistent inputs are not possibile, because the only way errors can be made is the options frame, but errors in this part are excluded by exceptions which advise the user of bad inputs, for example selection quota, filter accuracy greater 100% or string inputs.

To test the programm and the algorithm we selected 20 images (figure 1) from a random area in germany. These images were taken on various daytimes, seasons and weather conditions. For the first test we loaded five images into ASAP and have choosen a selection quota of 25%, so the result in the best case will consist of the five most different images.

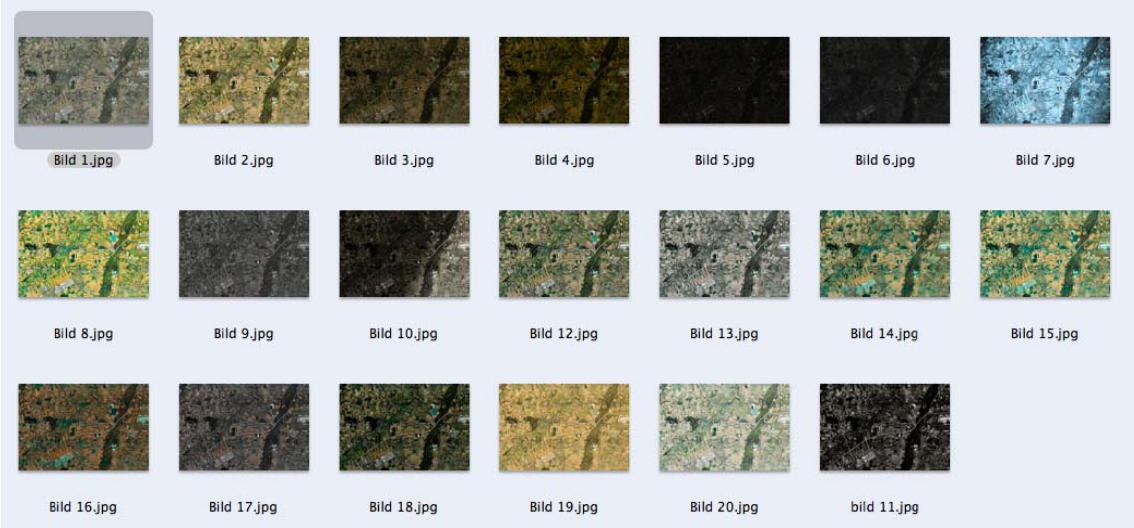

Figure 3: Overview of the test images.

Obtaining the output (figure 4) of the selection performed by ASAP, we see that the variety of the images is quite heterogeneous. Image 12 seems to be normal with an average brightness and an average constrast. Image two has, compared with the other images a high contrast and a higher average brightness; image three has a lower brightness than the average image and in some places a high contrast. Image four can be described as “normal” with a yellow undertone; image five has a low contrast and a low brightness and is probably a night image. Evaluating this result subjectively we can infer that that every kind of image properties of the input is represented.

In the next test we tried to find out if ASAP detects cloned images. For this test we cloned all the 20 images, loaded 40 images (the clones and the originals) into ASAP and chose a selection quota of 50% In the result we got some originals and some clones, but no image in the output was selected twice.

In order to verify the second test case we cloned the images a second time and filtered the images with a selection quota of 34%. In this test we had, similar to the test before, no duplicates, but the images were choosen of three sets. After performing the tests we conclude that ASAP will not have possible duplicates in the output as long as there are equal or less images in the output as unique images.

Figure 4: Selection results.