PROCESSING EXTENSION FOR THE SPECTRAL DATABASE SPECCHIO

Andreas Hueni [a], Mathias Kneubuehler [a], Jens Nieke [b] and Klaus I. Itten [a]

[a] University of Zürich, Department of Geography, RSL, Zürich, Switzerland - (ahueni, kneub, itten)@geo.uzh.ch
[b] ESA-ESTEC, Noordwijk, The Netherlands - Jens.Nieke@esa.int

KEY WORDS: spectral database, spectroradiometer, spectral space, data processing

ABSTRACT:

SPECCHIO is a spectral database combined with user-friendly interface software designed to store spectral data acquired by spectroradiometers and associated metadata. SPECCHIO was developed to support long-term usability and data sharing between researchers. The user interface focused on three main tasks: data input, data editing and data export.
Experience, however, has shown that users are interested in seeing spectral processing capabilities added to SPECCHIO. Such operations are to be applied to the data during data output, leaving the original data in the database intact. Typical examples are the removal of noisy wavelength regions, spectral convolutions to other sensors or statistical calculations such as mean or standard deviation. The main requirements of such processing capabilities are: (a) handling of different sensors and instruments used during data capture, (b) user configurable sequences of operations and (c) visualisation of results and processing progress.
A solution to the problem of such a configurable and generic processing, based on the concept of features spaces, has been implemented as part of the SPECCHIO software package and is included from version 2.0 onwards.

1. INTRODUCTION

Collection of spectral measurements by spectroradiometers is undertaken for two main reasons: (a) basic investigation of the relationship between physical or biochemical properties and the electromagnetic reflectance of objects and (b) calibration, validation and simulation of remote sensing imagery and its data products.
Although the use of spectroradiometers has become widespread in various fields of research, the management and processing of the resulting spectral data remains an issue largely untouched and only a few tools are available to help the researchers. The SPECCHIO system has been developed at the Remote Sensing Laboratories (RSL) with a focus on data management. The spectral data are at the centre of the data model and are supported by a host of metadata that ensure the long-term usability and shareability (Hueni et al., 2009).
During design, special attention was paid to keep the data input mechanism as automated as possible and to offer the user the option to update metadata of several records with a singular operation. Consequently, the system is capable of loading large numbers of spectra in short time, described by a considerable number of auto-generated metadata.
The SPECCHIO data schema bases on a MySQL database (MySQL AB, 2005) and the end user application is written in Java (Sun Microsystems Inc., 2006). The SPECCHIO system can therefore be operated in heterogeneous computing environments, offering multiuser access to a centralized database and enabling easy data sharing within and even across research groups.
The need for a processing extension arose due to three main reasons: (a) some users would like to apply some simple functions to the data before exporting to a file for subsequent use, e.g. mean value calculation, (b) some calculations should usually be applied before exporting the data, like application of reference panel correction factors and (c) various calculations may be needed to develop information based on data stored in the database, which may be too complex or cumbersome to carry out without the full spectral/metadata information context available within the system, e.g. the retrieval of BRF from goniometer measurements (Hueni et al., 2008).

2. REQUIREMENTS

2.1 Structure

The structure of processing operations applied to data can be described by a directed graph, as can be observed in Figure 4. Such a network consists of processing modules and data sources/sinks. Data is read from the sources by the modules connected to them, transformed within the modules and written to the connected sinks.
Such a modular structure provides high flexibility: even complex processing sequences can be built by connecting basic components (Hueni and Tuohy, 2006). The use of modules also helps to minimize the code redundancy.

2.2 Data Integrity

SPECCHIO has been designed as a repository for spectral data. Processing components are not to modify the original database contents. However, it would be possible for components to create new database entries in the system. In any case, full reprocessing functionality must be guaranteed, i.e. identically configured processing networks must produce the same results with every run.

2.3 Interactivity and Visualisation

Design and configuration of processing networks should be possible in two main ways: (a) as an interactive procedure, applicable by the end user and (b) as an internal API (Application Programmers Interface) that allows the

definition of either complex or more often used processing sequences programmatically.
The network, the processing progress and the changes applied to the data should be presented to the user in a visual manner, giving a comprehensive overview of the processing at all times.
Monitoring of both intermediate and final products of the processing network must be possible by either visual output or by output to files. This enables the user to check if the processing is having the desired effect on the data.
The processing must not influence the responsiveness of the system, allowing the execution of time-consuming computations while the user can continue to work. Several instances of the processing component should be able to co-exist without influencing each other.

2.4 Generic Design

The SPECCHIO database supports a variety of spectroradiometers. In fact, any sensor that produces point measurements related to wavelength can be modelled in SPECCHIO.
Processing modules should not be targeted at a specific sensor but written as generically as possible. However, care must be taken, as not to mix data of different sensors in an uncontrolled manner and the system must provide according functionality. From a stricter point of view, the mixing of data is to be controlled even more extensively: data captured by differing instrument of the same sensor model are not to be mixed before inter-calibration factors have been applied. Depending on the instruments, the same may even apply for spectra acquired by the same instrument but with different instrument calibrations. Finally, the measurement unit (digital number, radiance, absorbance, transmission or reflectance) further defines what measurements may logically be combined.

3. CONCEPTS

3.1 Space Processing Network

An analysis of the structure of processing networks in the context of spectral processing shows that they are formed by two type of components connected by edges: processing modules and data sinks/sources. The modules are effecting the transport of data from sources to sinks, modifying them along the way. These data sources/sinks need to be able to hold collections of spectra acquired by different instruments and the number of data points per spectrum may change as the data progresses through the network.
A solution to the generic, flexible requirements outlined above is given by the concept of the 'Space Processing Network'. It is based upon the definition of feature spaces by Landgrebe (1997). The continuous, spectral response of an object is transformed into a discrete space by the sampling instrument. This transformation is defined by the sensor characteristics (Hüni et al., 2007a). Thus, data captured by different instruments are contained by differing spaces. A space has a dimensionality equal to the number of bands of the used sensor and the spectra are data vectors contained within this space.
Within the SPECCHIO design, spaces are therefore containing the following information: number of dimensions, wavelength per dimension, collection of all data vectors and a reference to the database spectrum record for every data vector. The database reference ensures that the full context of every spectrum can be retrieved from the database by a processing module if required while keeping the memory footprint of the space to a minimum.
Processing modules are effecting a transformation on a space, i.e. the spectral data vectors of the input space are transformed to an output space. The algorithm of the processing module defines the dimensionality of the resulting space. This is illustrated in Figure 1 with an input space of dimensionality N being transformed into another discrete space of dimensionality M. Although processing modules tend to have singular input/output in most cases, they may have multiple inputs and generate multiple outputs.

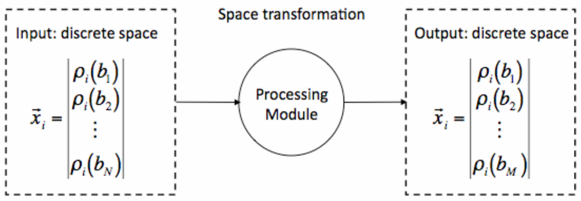
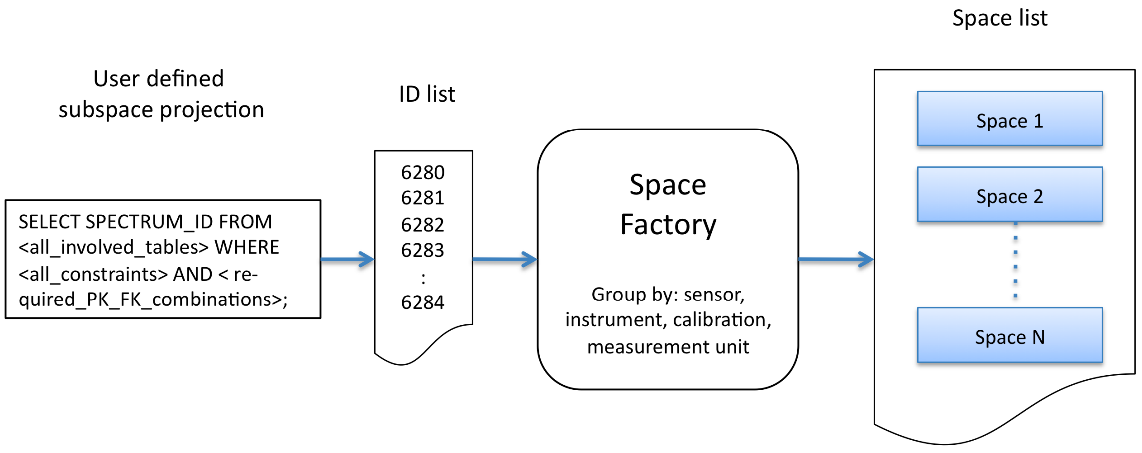

Space transformation
Input: discrete space
$\vec{x}_i = \begin{vmatrix} \rho_i(b_1) \\ \rho_i(b_2) \\ \vdots \\ \rho_i(b_N) \end{vmatrix}$
Processing Module
Output: discrete space
$\vec{x}_i = \begin{vmatrix} \rho_i(b_1) \\ \rho_i(b_2) \\ \vdots \\ \rho_i(b_M) \end{vmatrix}$

Figure 1. Transformation into a new space by a processing module

3.2 Space Factory

The generation of spaces can be triggered in two principal ways: (a) based on a query by the user and (b) as a result of the transformation of an input space by a processing module.
The Space Factory is a conceptual, central component of the SPECCHIO system. It creates new spaces based on given inputs and contains the logic to form 'non-mixed' spaces.
Spaces are used throughout the system for processing, visualisation and file output. In all these cases, vector data must be related to spectral dimensions and this information is held by the space. Moreover, a space can hold only spectra that are of the same dimension. It is the job of the Space Factory to create such spaces.
Assume the use case of displaying spectral plots of a number of spectra. In a first step, the user will select the spectra to be plotted by effecting a subspace projection (Hüni et al., 2007b). Internally, this will yield a number of record id's that are matching the user's selection. These ids are now handed to the Space Factory. Internally, spaces are created for all existing combinations of the respective sensors, instruments, calibrations and measurement units associated with the spectra (see Figure 2).



User defined subspace projection
SELECT SPECTRUM_ID FROM <all_involved_tables> WHERE <all_constraints> AND < required_PK_FK_combinations>;
ID list
6280
6281
6282
6283
:
6284
Space Factory
Group by: sensor, instrument, calibration, measurement unit
Space list
Space 1
Space 2
Space N

Figure 2. Building spaces based on user defined subspace projections

The Space Factory returns a list of the created spaces. Each space can now in turn be used as an input argument of a plotting class instance. Utilizing the Space Factory ensures that all spectra contained by a space have a common wavelength per band and the same measurement unit, i.e. the following processing modules do not need to carry out uniformity checks but can apply their algorithms directly, e.g. plotting of spectral vectors against the common wavelengths of the space.

3.3 Processing Flow Control and Synchronisation

This sub-concept provides a solution to the tricky problem of triggering the processing of data by the single modules.
The network structure of the processing modules and spaces implies that a sequential flow control is cumbersome to be employed. A process with several inputs can only be started when all the input spaces are ready, i.e. the modules preceding the input spaces must have finished their processing as well. Managing these dependencies with a central controlling instance would require the analysis of the topography of the underlying network. Furthermore, some processes may also be run in parallel and the requirements state that the application should not be blocked during processing.
The chosen solution utilises the fact that the network structure defines the order of processing. Spreading the runtime behaviour over all network components eliminates the need for centralized control. The mechanism is quite simple: a processing module needs to wait till all input spaces are ready and then can start processing. Such behaviour can be implemented by using threads and monitors (Hoare, 1975). Every processing module is implemented as an individual thread, ensuring that the system remains responsive during processing and parallel processing capability is provided. Threads are calling a ‘data ready’ method on all input spaces in order to check of the processing can be started. The use of monitors to guard the ‘data ready’ method of the spaces ensures that a thread is waiting (i.e. sleeping) till the space changes the status to ‘data ready’ and signals all waiting threads accordingly.
Processing is started by forcing the starting space to load its spectra from the database. The loading ends with the status of the space being set to ‘data ready’ and waking the threads waiting on the space’s monitor. Consequently, the whole processing network will gradually be executed with the processing flow being implicit part of the network’s components behaviour.

3.4 Graphical Representation

The graphical representation of the processing network was separated as much as possible from the processing mechanisms. For this reason, an existing, specialized visualisation package was employed. JGraph is a freely available, Java Swing User Interface compliant component for the visualisation of graphs (JGraph Ltd, 2008).
Both spaces and processing modules hold a reference to a GraphCell instance, which is the base class of all vertices within the JGraph package. These cells are then placed on a graph component and automatically drawn and updated. The graph itself is held by a ProcessingPlane instance. The ProcessingPlane is an object that acts as a container for all spaces and processing modules. The logical and graphical connections of these components are however kept separated, i.e. an edge between two vertices is only a graphical representation of the internal logical link and is not used for the processing. Thus, it would be feasible to build a working processing network without any graphical representation.

4. RESULTS – A CASE STUDY

This section employs a case study to demonstrate the implemented processing system and serve as a proof of concept.
Reflectance is one of the basic quantities that are regularly measured in the remote sensing context. It proves useful as it normalizes the reflected energy to the irradiance and thus eases the comparison of signatures taken under varying illumination conditions. A common technique to acquire reflectance is based on taking reference readings from a reference panel (Pfitzner et al., 2005). Ideally, such a panel would be 100% reflective over all wavelengths. However, it is a technical challenge to engineer such a material. The well know Spectralon panels are delivered with calibration reports stating the spectral reflective properties (for a plot see Figure 3) (Labsphere Inc.).

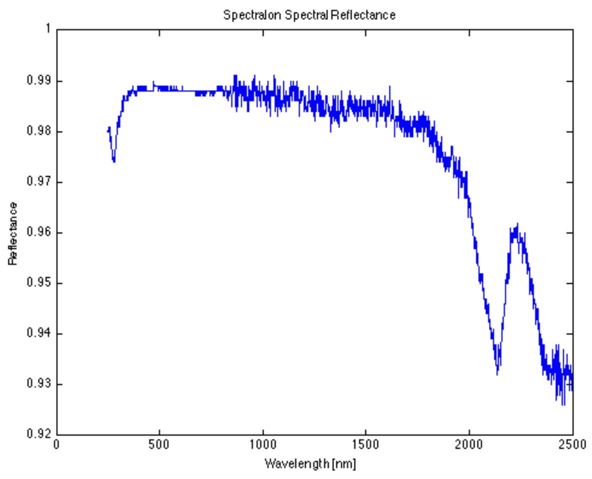


Figure 3. Spectralon Spectral Reflectance

The acquired reflectance spectra can be correct by post-processing if the non-idealness of the used panel is known. In the case of radiance measurements of both target and reference panel, the corrected reflectance is calculated by:

$$\rho = \frac{L_{tar}}{L_{ref}} \rho_{ref} \qquad (1)$$

The conversion of radiance to reflectance including the panel correction is expected to be a quite common processing step. However, the application of the correction factors tends to be cumbersome when applied manually every time. Furthermore, manual correction would be exceedingly tiresome when for a given collection of spectra more than one panel or more than one calibration set per panel applied.

The conversion process can easily be modelled as a processing network containing a number of processing modules and spaces (see Figure 4). The input space is shown in the upper left corner of the graph. It contains 66 vectors and has a dimensionality of 2151 bands. Note that the spaces are consecutively numbered in the upper left of the respective boxes. The first module 'Radiance to Reflectance' is building a ratio of the target and reference radiances:

$$\rho = \frac{L_{tar}}{L_{ref}} \qquad (2)$$

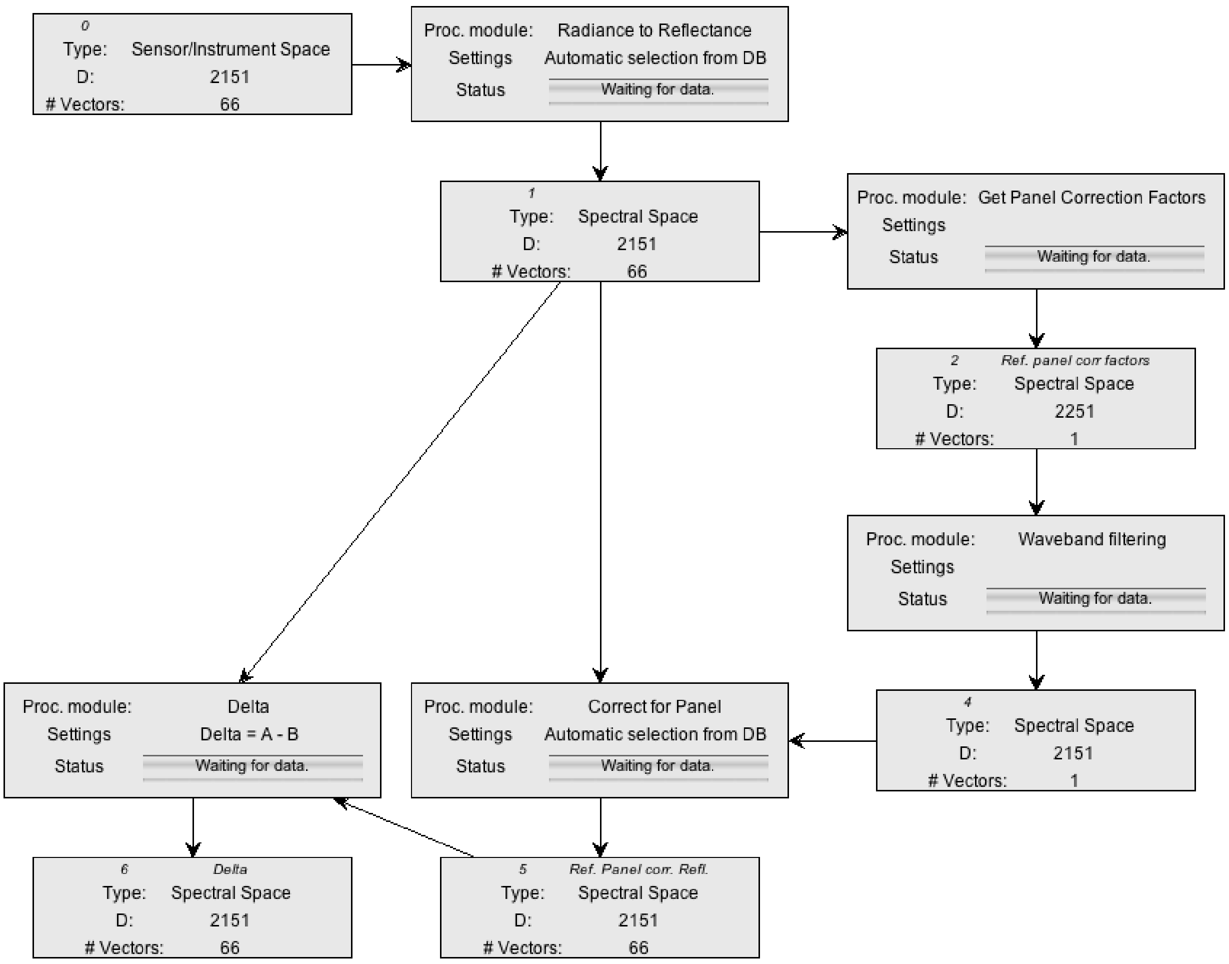


Figure 4: Processing network of the case study showing a conversion of radiance to reflectance including a correction for the reference panel non-idealness

The selection of the reference reading for each target measurement is fully automatic as the according information is already contained in the metadata. For an explanation of how such links are created please refer to Hueni et al (2009). The output of the first module is a further space (no 1). It still contains 66 vectors, i.e. spectra, but these are now reflectances.

Space number 1 serves as input for the module 'Get Panel Correction Factors'. This module uses metadata queries on the database to select the correction factors of the used panels that apply for the time of data capture. This results in a new space number 2 holding one vector of dimensionality 2251. This dimensionality results from the Spectralon calibration data where the reflectance of the panel is specified from 250-2500nm in 1nm steps. The correction factors cannot be applied to the spectra directly because of the dimensional mismatch. Therefore, a 'Waveband filtering' module is attached to space 2 and configured to cut wavelengths between 250 and 350nm. This results in a space number 4, whose dimensions are now tallying with the spectra in space number 1.

Spaces 1 and 4 are used as input for the module 'Correct for Panel'. The module ensures that the correct factors are applied to the spectra even in the case of multiple panels being used during data capture. The corrected reflectances are contained in space number 5.

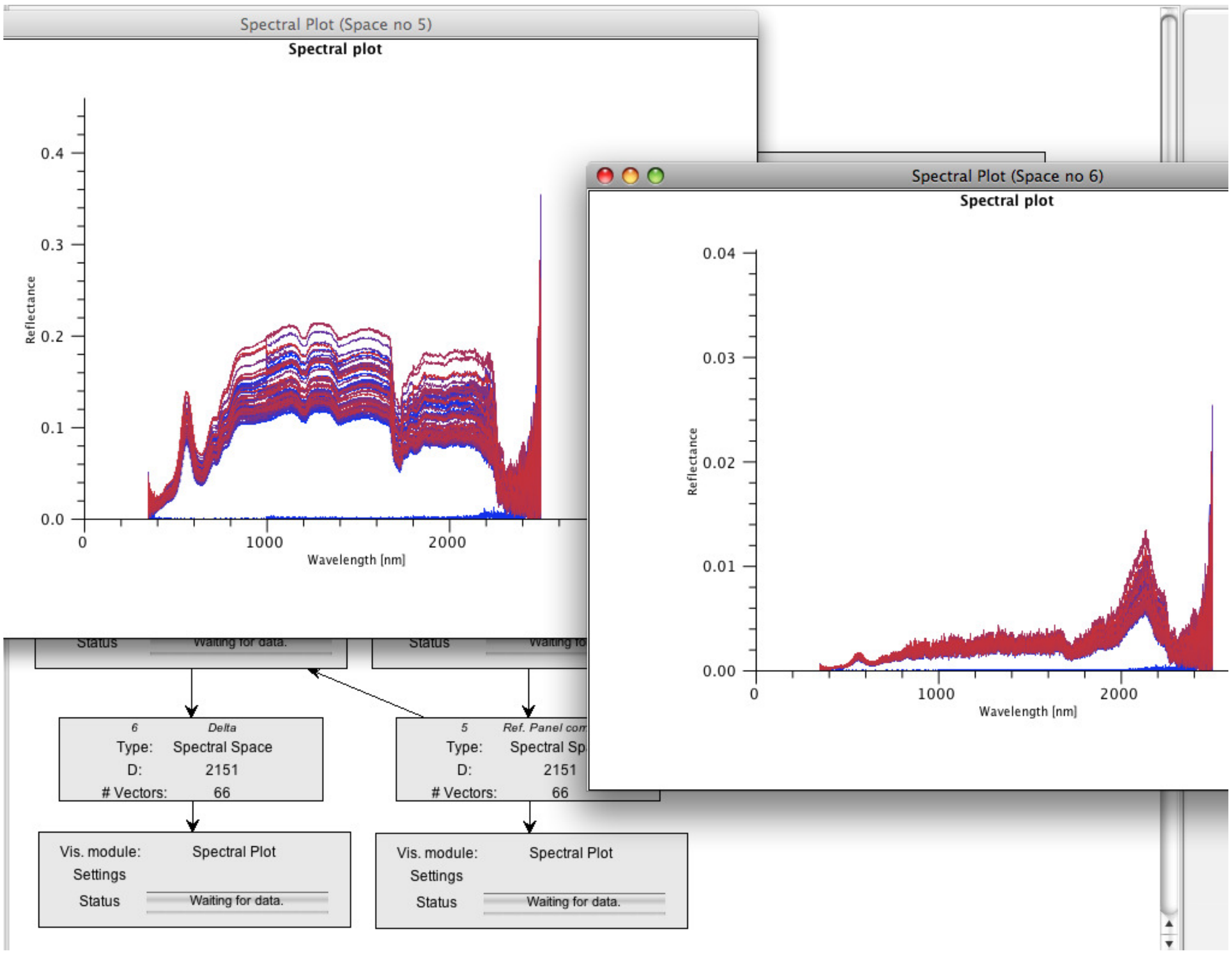
Finally, the module 'Delta' calculates the difference between corrected and uncorrected reflectances. The result of the correction procedure can be assessed visually by adding visualisation modules to the network (see Figure 5 showing spectral plots of the spectra contained by spaces 5 and 6).

5. CONCLUSIONS

The provision of processing capabilities within spectral database software packages is one of the features that many users would like to see. Apart from direct end user requirements, processing extensions are also of interest for the generation of information based on data stored in the database, such as the provision of BRF data retrieved from goniometer datasets (Hueni et al., 2008).

The technical requirements of such a processing feature are quite demanding, as a generic framework must be developed that can handle the data stemming from heterogeneous sensors and instruments and offer the possibility of interactive definition and configuration of processing operations.

It could be demonstrated that such a system can be designed and implemented successfully based on the concept of feature spaces and by utilising the multithreading capability of Java to control the processing flow while guaranteeing parallel processing and retaining system responsiveness. The processing extension will be integral part of the SPECCHIO application and will be featured starting with version 2.0.



Status Waiting for data.

Status Waiting fo

6 Delta
Type: Spectral Space
D: 2151
Vectors: 66

5 Ref. Panel cor
Type: Spectral Sp
D: 2151
Vectors: 66

Vis. module: Spectral Plot
Settings
Status Waiting for data.

Vis. module: Spectral Plot
Settings
Status Waiting for data.

Figure 5: Processing network with added visualisation modules and according output plots

RSL maintains an online version of the SPECCHIO database and interested researchers can acquire a database account for testing and data sharing purposes. For further information, please visit to the SPECCHIO website (www.specchio.ch).

6. REFERENCES

Hoare, C. A. R., 1975. Monitors: An Operating System Structuring Concept. Communications of the ACM 17, 549-557.

Hueni, A., Nieke, J., Schopfer, J., Kneubühler, M., Itten, K., 2009. The spectral database SPECCHIO for improved long term usability and data sharing. Computers & Geosciences 35(3), 557-565.

Hueni, A., Schopfer, J., Schläpfer, D., Kneubuehler, M., Nieke, J., 2008. PRE-PROCESSING OF DUAL-VIEW FIGOS DATA: TOWARDS OPERATIONAL BRDF RETRIEVAL. In: Proceedings ISPRS, Beijing.

Hueni, A., Tuohy, M., 2006. Spectroradiometer Data Structuring, Pre-Processing and Analysis - An IT Based Approach. Journal of Spatial Science 51(2), 93-102.

Hüni, A., Nieke, J., Schopfer, J., Kneubühler, M., Itten, K., 2007a. 2nd Generation of RSL's Spectrum Database "SPECCHIO". In: Proceedings Schaepman, M. E., Liang, S., Groot, N. E., Kneubühler, M. (Eds.), 10th Intl. Symposium on Physical Measurements and Spectral Signatures in Remote Sensing, Davos (CH), XXXVI, Part 7/C50, pp. 505–510.

Hüni, A., Nieke, J., Schopfer, J., Kneubühler, M., Itten, K., 2007b, 23-25 April 2007. Metadata of Spectral Data Collections. In: Proceedings 5th EARSeL Workshop on Imaging Spectroscopy, Bruges, Belgium, pp. 14.

JGraph and JGraph Layout Pro User Manual, 2008. Version JGraph Version 5.12.0.0th, JGraph Ltd, Northampton, UK, 140 pp.

Labsphere Inc., North Sutton, NH, USA.

Landgrebe, D., 1997. On Information Extraction Principles for Hyperspectral Data, Purdue University, West Lafayette, IN, 34 pp.

MySQL, 2005. MySQL AB.

Pfitzner, K., Bartolo, R. E., Ryan, B., Bollhöfer, A., 2005. Issues to consider when designing a spectral library database. In: Proceedings SSC 2005 Spatial Intelligence, Innovation and Praxis: The national biennial Conference of the Spatial Sciences Institute, Melbourne, pp. 416-425.

JavaTM 2 Platform Standard Edition, 2006. Version 5.0, Sun Microsystems Inc., Santa Clara, CA.