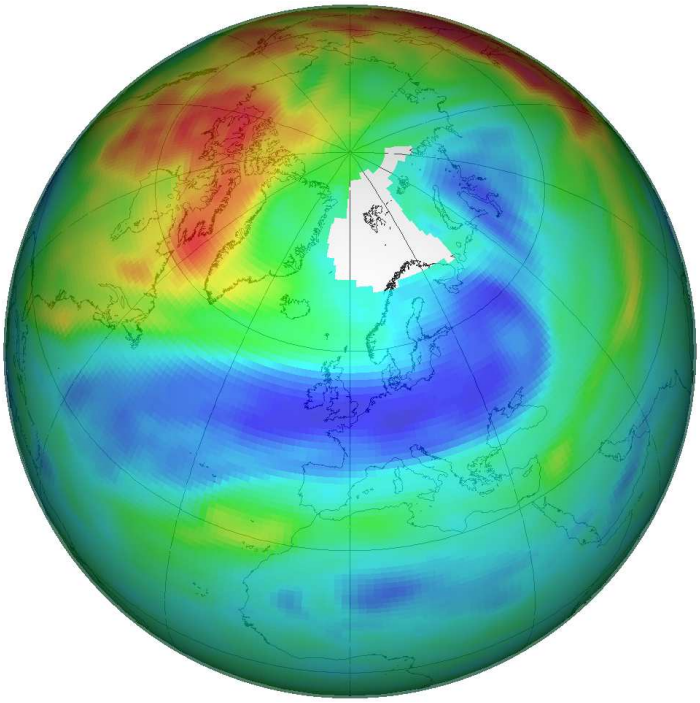
Basic Envisat Atmospheric Toolbox



Tutorial

Basic Envisat Atmospheric Toolbox Tutorial

S. Cadot
S. Niemeijer
M. de Graaf

December 2004

Front page picture: Mini-ozone hole over Europe of 1 December 1999, based on KNMI GOME Fast Delivery data visualised with VISAN.

Contents

BEAT: An introduction

BEAT, the Basic Envisat Atmospheric Toolbox, is a set of libraries and programs that provide access to GOME, GOMOS, MIPAS and SCIAMACHY data-products in a variety of ways.

A short history of BEAT

Prior to the introduction of BEAT, atmospheric scientists who wanted to use remote sensing data for research purposes often needed to dedicate a significant effort to the implementation of data ingestion routines, based on (sometimes incomplete) specifications. This work would usually be duplicated by different institutions, universities, or even individual scientists.

With the inception of the Envisat platform, this already difficult situation threatened to become a real problem. Envisat carries 10 scientific instruments, most of which exceeded their predecessor instruments both in complexity and in the amount of data produced. In order to cope with this increased complexity, ESA mandated that all Envisat data products should follow a common format (the Payload Data Segment or 'PDS' format).

While this was a very good idea that prevented an enormous number of heterogeneous and incompatible formats from springing into existence, it also placed a serious burden on both the producers and consumers of Envisat data products. On the 'consumer' side (i.e., the scientific community), it raised the software-engineering complexity of data ingestion beyond a level that the individual scientist could reasonably be expected to cope with.

In recognition of this fact, ESA decided to fund the development of three 'software toolboxes' that would provide functionality to do ingestion and basic analysis and visualisation: BEAM (Basic ERS/Envisat (A)ATSR and MERIS Toolbox), BEST (Basic Envisat SAR Toolbox) and BEAT: the Basic ERS/-Envisat Atmospheric Toolbox. These toolboxes are distributed as open source software, meaning that source-code is available to end-users, who are allowed (in fact, encouraged) to improve the functionality of the software.

The BEAT software has been available for three years now, and is still constantly being improved based on new insights and user feedback. In its current form, BEAT consists of three major parts:

- BEAT Layer 1 provides access to each and every byte of data contained in Envisat and GOME data products, from within `C`, `FORTRAN`, `IDL` and `MATLAB`.
- BEAT Layer 2 provides a simplified but powerful interface to the main

scientific data contained in each of the data products. It can be used from `C`, `FORTRAN`, `IDL`, `MATLAB` and `Python`.

- The `VISAN` application combines the popular open source scripting language `Python` with the power of BEAT Layer 2 in one integrated GUI environment, providing advanced analysis and interactive visualisation capabilities.

About this tutorial

This tutorial provides an introduction to BEAT and its capabilities for use in atmospheric science. It considers a number of real-life scientific case studies. As such, it shows how one proceeds from asking a scientifically interesting question to answering it, using the tools provided by BEAT.

The first case is an example of the simplicity for end-users of BEAT's data ingestion capabilities and the powerful GUI environment that `VISAN` provides. With only two lines of code and a few intuitive manipulations, global assimilated ozone fields from GOME covering an entire month can be loaded and visualised in a sequence of 3D pictures.

The second chapter shows how BEAT layer 2 can be used to load and compare ozone profile data from GOMOS and MIPAS, without any knowledge of the data-products, using `IDL`. Within `IDL` the user can access the data, manipulate and plot it. In this example MIPAS ozone profiles are compared with a GOMOS ozone profile, after selection of co-located data and conversion of units.

The third case shows how to use the Polarisation Measurement Device (PMD) channels from SCIAMACHY to make RGB pictures from the Earth's surface. Although SCIAMACHY is a spectrometer, with a poor spatial resolution compared to imagers, RGB pictures from this sensor can be very helpful for cloud detection, guidance of the human eye, first verification of geolocation information, etc. The PMD channels have the best spatial resolution and can be used for this. To access the right data a thorough knowledge of the SCIAMACHY data structure would be necessary. The tool provided with this tutorial guides the user through the necessary steps, explaining everything in detail and showing the capability of BEAT layer 1 to access all the data.

The last case compares GOME reflectances with SCIAMACHY reflectances. GOME and SCIAMACHY are both spectrometers, with a broad spectral range and a high spectral resolution, making them unique instruments for the study of the Earth's atmosphere. In-flight calibration is crucial to these space-borne sensors, which can be done by inter-comparison with independent data sources like other sensors. Although GOME and SCIAMACHY have different spatial and spectral resolutions, they fly in the same orbit, half an hour apart, which makes them very suitable for a comparison study.

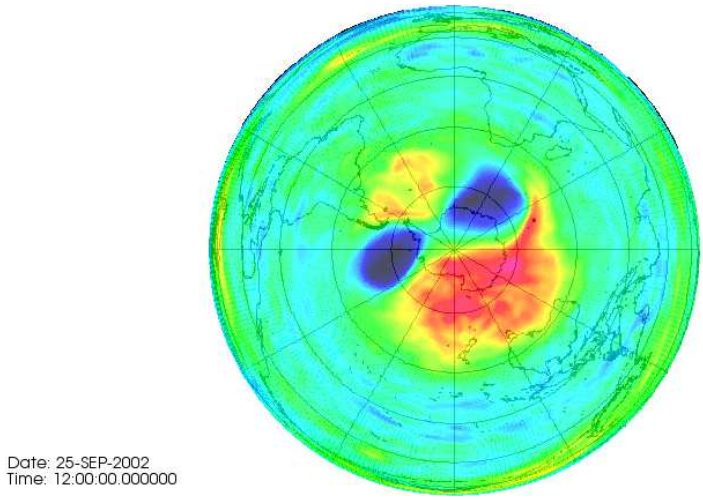
The case studies in this tutorial are specifically designed examples or based on earlier work on verification of SCIAMACHY data products performed at KNMI. Except for the first example, the `IDL` interface is used like in the original routines. Both BEAT layer 1 and layer 2 routines are used. By showing different ways of using BEAT, it is our hope that this document will help atmospheric scientists to decide if this toolbox is the right tool for their work, and to ease the learning curve.

Further information

BEAT consists of an elaborate set of libraries, command line tools, and a GUI-based visualisation and analysis application (`VISAN`). This tutorial document only scratches the surface of the many ways in which BEAT could be used to do science. If you need a more in-depth look or an authoritative reference, your first source of information is the documentation that comes with BEAT, the most current version of which is available from `http://www.science-and-technology.nl/beat/`.

For some issues, the best source of information is provided by the BEAT developers themselves. Feel free to contact them by e-mail (contact information can be found on the BEAT website).

1 Ozone map from GOME L4 data.



1.1 The problem

In April 1995, GOME was launched on-board the ERS-2 satellite with the objective of determining the global distribution of ozone, and several other trace gases which are important for the ozone chemistry in the Earth's stratosphere and troposphere. From it's launch until June 2004, GOME has been constantly monitoring the variations of the 'ozone layer'. Daily global measurements are available on the Internet at the Tropospheric Emission Monitoring Internet Service (TEMIS) website (`http://www.temis.nl`). In September 2002 predictions of an early split up of the 'ozone hole' over Antarctica received world wide attention. The ozone hole appears over the cold region of Antarctica when the sun returns after the polar winter - strong ozone depletion occurs with the combination of very low temperatures and high solar radiation. The ozone hole is surrounded by a vortex of strong winds that stops air moving between polar regions and warmer mid-latitude regions, keeping the hole in the ozone stable.

In this exercise GOME level 4 data, downloaded from the TEMIS website, will be used to produce a sequence of plots in `VISAN` showing this early split up of the ozone hole.

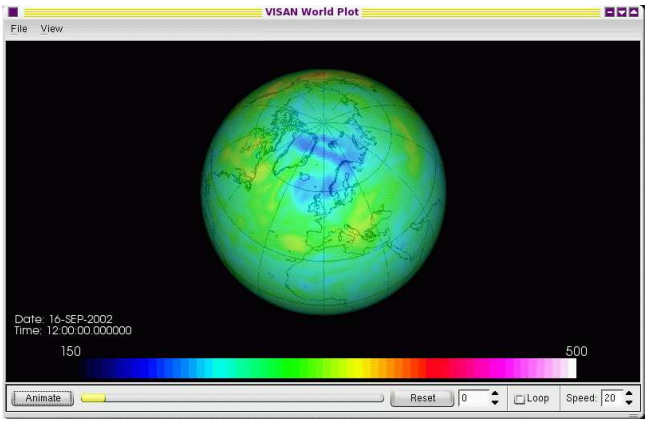


Figure 1.1: *The* VISAN *world plot window.*

1.2 The work that needs to be done

- The assimilated level 4 GOME data, provided locally, needs to be read.
- Plots must be created using a colour table displaying the ozone values overlaying a globe or world map.
- The plots must be displayed after one another to produce a sequence or movie.

1.3 Overview of the VISAN program

A `python` script `ozone_map.py` is provided in the directory `ozone-map`, which will help the user perform the steps of the previous section. Here some extra information is given.

To start the tutorial, go to the `ozone-map` directory and start `VISAN`. Load and execute the script by selecting `'Load and Execute Script...'` from the `File` menu (or type `CRTL+O`) and select `ozone_map.py`.

That's all there is to it. A window will open like the one in Figure 1.1. Examination of the script will show that it consists of only two lines:

```
>>> data = beatl2.ingest('tutorial-data/ga*.gom','product_type=GOME_L4')
>>> wplot(data, colortable="Ozone", colorrange=(150.0, 500.0))
```

The first line loads the data, using the BEAT command `beatl2.ingest`. This command can be used for ingestion of all data that is supported by BEAT Layer 2. The first argument of this command is the `filepath` parameter, telling it where it can find the data. Other arguments are filter options, in this case to tell BEAT that it is to expect GOME level 4 (assimilated) data. For more information see the BEATL2-Python documentation.

The second line opens the `VISAN` world plot window, showing the data in the default setting, usually a 3D picture like Figure 1.1. The last two arguments of this command tell it to use a proper colour table and plot range. Of course, this can also be changed interactively from the plot window. The world plot window

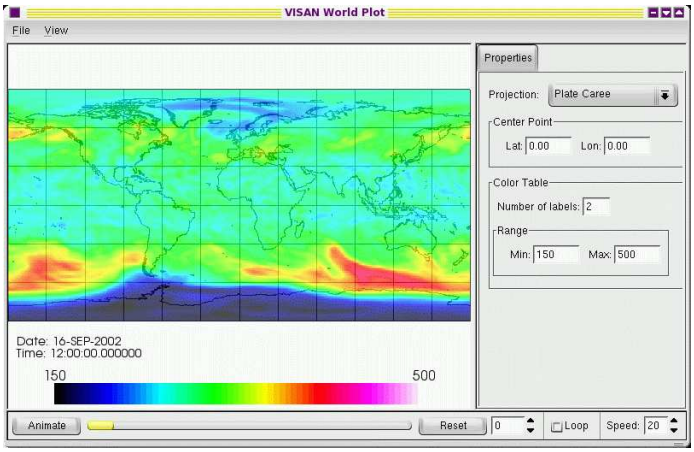


Figure 1.2: GOME *L4 ozone map, using a Plate Caree projection.*

has a `Properties` tab, where the properties of the plot can be changed. It can be selected from the `View` menu. From the `Properties` tab the projection type can be changed, e.g. for those more at ease with a world map, instead of a 3D plot, a plate caree projection might be more appropriate, see Figure 1.2.

Using the `properties` tab, change the projection to Lambert azimuthal, set the central point to -90.0 latitude and 0.0 longitude and press `Animate`. The ozone hole over Antarctica will rotate, split up, fill up and return again.

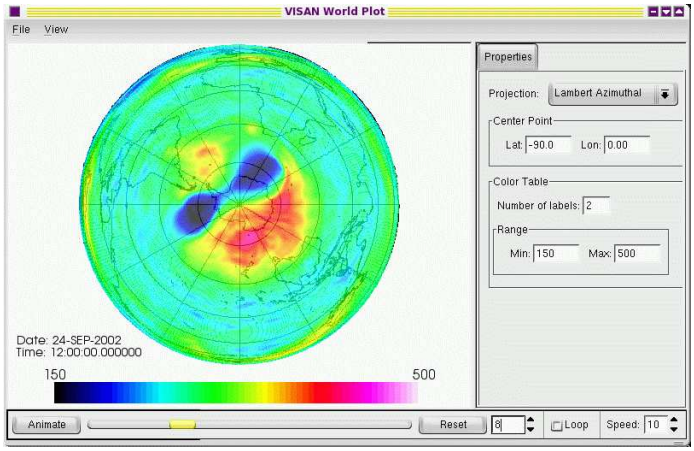


Figure 1.3: *Still of the ozone distribution animation. Shown is the split up of the ozone hole over Antarctica at 24 September 2004.*

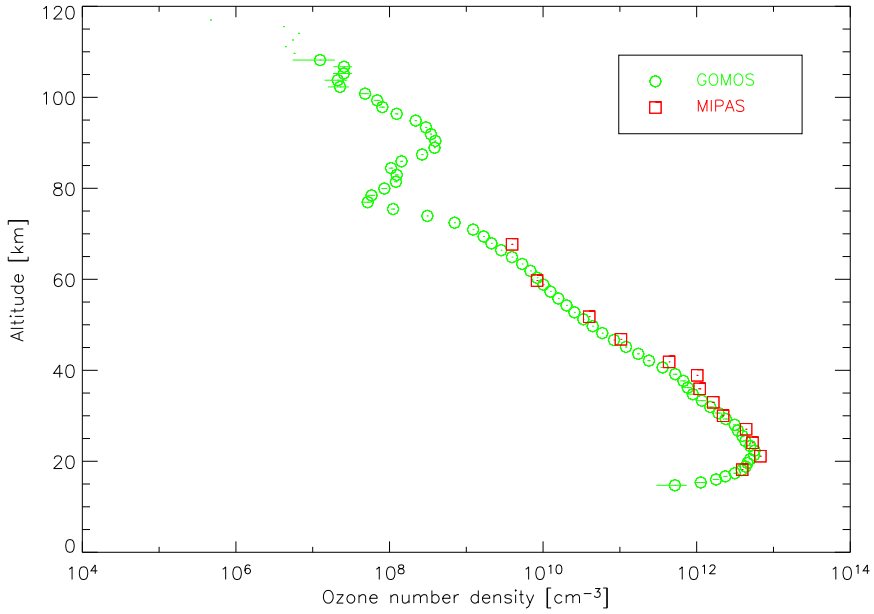
1.4 The BEAT routines

This was a very simple example of the ease with which BEAT-II handles data in an intuitive manner. With only two lines of code, data can be visualised and manipulated using user-friendly GUI software. In this example the only BEAT routine used was

```
BEATL2.INGEST()
```

In the next chapters more advanced uses of BEAT will be demonstrated, using all kinds of atmospheric data products. Although more knowledge of the data products is required to understand what can be done with it, the use of BEAT is always the same: a straightforward approach to loading the data in the most intuitive way. The same BEAT commands can be used in `IDL` and `MATLAB` as in `VISAN`, but notice that the dot (`.`) in the commands is replaced by an underscore (`_`) in these languages.

2 Ozone profiles from GOMOS and MIPAS



2.1 The problem

On-board the Environmental Satellite ENVISAT are three instruments that can measure ozone: GOMOS (Global Ozone Monitoring by Occultation of Stars), MIPAS (Michelson Interferometer for Passive Atmospheric Sounding) and SCIAMACHY (Scanning Imaging Absorption SpectroMeter for Atmospheric CartographY). All instruments have different characteristics and different purposes; e.g. GOMOS was specifically designed to measure ozone using star occultation, while MIPAS was designed to measure high-resolution gaseous emission spectra at the Earth's limb in the near- to mid-infrared (IR) and SCIAMACHY was designed to measure entire reflectance spectra from the UV to the far IR. It is interesting, however, to inter-compare the ozone profiles from the various instruments, because they all are on the same platform, so potentially the various

instruments will be scanning air masses close together. Only, GOMOS and MIPAS are backward scanning instruments, where SCIAMACHY is scanning forward in limb mode.

In this module ozone profiles of MIPAS and GOMOS will be sought that are sufficiently close together and compared with each other, using BEAT layer-II software and GOMOS and MIPAS level 2 data.

2.2 The work that needs to be done

In order to compare ozone profiles, the following steps need to be taken:

- The level 2 data from both instruments must be read.
- The data must be scanned to find co-located data, i.e. only data that are sufficiently close together can be used for the comparison. Of course, 'sufficiently' is very subjective and will vary according to the application, availability of data and needed accuracy. Here we will compare any MIPAS data that is closer than 200 km from the measurements from GOMOS.
- In order to compare the data, the ozone profiles of MIPAS given as number densities, must be converted to mixing ratios, the unit of the GOMOS ozone profiles.

2.3 Overview of the IDL program

The IDL program `ozone_profile.pro` (in the directory `O3-profile`) will guide the user through the steps of the previous section. Here some background information will be given about the program.

To start the tutorial program, go to the `O3-profile` directory and start `IDL` or `IDLDE`. From the `IDL` prompt type

```
IDL> ozone_profile
```

or in `IDLDE` load the file `ozone_profile.pro`, compile (`CTRL+F5`) and run (`F5`) it.

The user is asked to load a GOMOS file. This must be a level 2 file. A sample file is provided, but any other GOMOS level 2 file containing ozone profile measurements will do. After the file is selected, the ozone profile data is loaded. This can be done with the BEAT II command `beatl2_ingest()`:

```
IDL> gomosdata = beatl2_ingest(gomosfile,'species=o3')
```

Here, `gomosfile` is the file that was just selected and `'species=o3'` the filter for the `beatl2_ingest()` command. See the BEAT-II data description for other filters. Obviously, this filter will make sure the ozone data of the GOMOS file is loaded.

The data is loaded into a standard (anonymous, or unnamed) `IDL` structure, containing the ingested BEAT-II record. Type `HELP, gomosdata, /STUCTURE` to view the contents of this structure. It is a BEAT-II record, a description of which can be found in the BEAT-II data description, (under the link `GOM_NL__2P_profile`). Note that this record contains all the ozone data, with

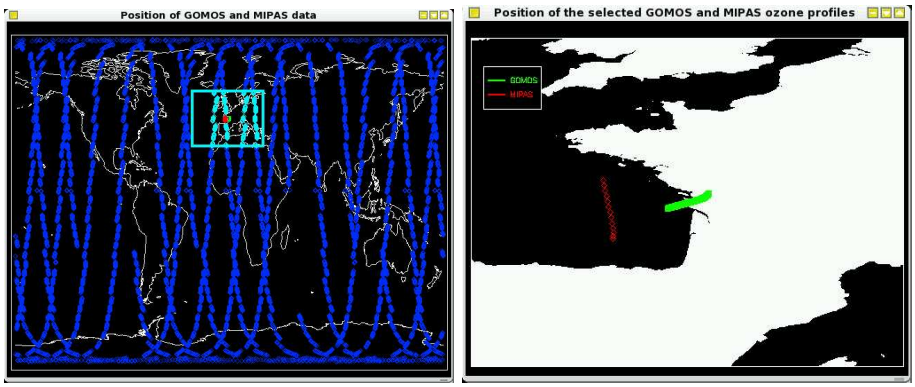


Figure 2.1: *Left panel: World plot with* GOMOS *data location (green) and* MIPAS *data locations (blue and cyan and red). Right panel: Location of the selected, co-located,* GOMOS *and* MIPAS *ozone profiles.*

time, longitude/latitude, altitude and some more information. We can directly see where, when and how this information was acquired. The location of this GOMOS ozone profile, e.g., is shown on a world map.

Next, we need some MIPAS data to compare to our GOMOS profile. Since we do not know a priori at which location the GOMOS profile is taken, we may start with an entire day of MIPAS data and select suitable profiles from that. The user is asked to specify a directory containing MIPAS level 2 data. A directory with sample files is provided, but any MIPAS level 2 files containing ozone profiles will do. The sample files are from the same day as the sample GOMOS file. All the ozone data in all these files can be ingested with the same command as with the GOMOS data:

```
IDL> mipasdata = beatl2_ingest(filenames,'species=o3,include_nan=1')
```

The `include_nan` filter option will be explained later and `filenames` is now a string containing all the MIPAS filenames. This single command will load all ozone data from all the files into a BEAT-II record, a description of which can again be found in the BEAT-II data description (under `MIP_NL__2P_profile`). Note that the two data-records from both GOMOS and MIPAS have the same generic structure, only the lengths of the records differ.

If all the files of one day were loaded, it will look something like the blue data in the left panel of Figure 2.1: the MIPAS data of one day. This will be too much too use for a comparison. One way to reduce the amount of data is to use additional filter option in the `BEAT_INGEST()` command. If, e.g., a geographical filter is used allowing only data that was retrieved within 30° longitude and 15° latitude from the GOMOS profile, only the data depicted in cyan in Figure 2.1 would have been loaded.

However, we will use the BEAT function `beat_find_colocated_data()` to select a data-subset that is sufficiently close to the GOMOS data. This function takes five arguments: the first two arguments are datasets from which co-located data should be found. The last three arguments are limits within which the co-location must fall, the first limit being time (in seconds), the second distance (in kilometers) and the third altitude (in kilometers). So when we want only

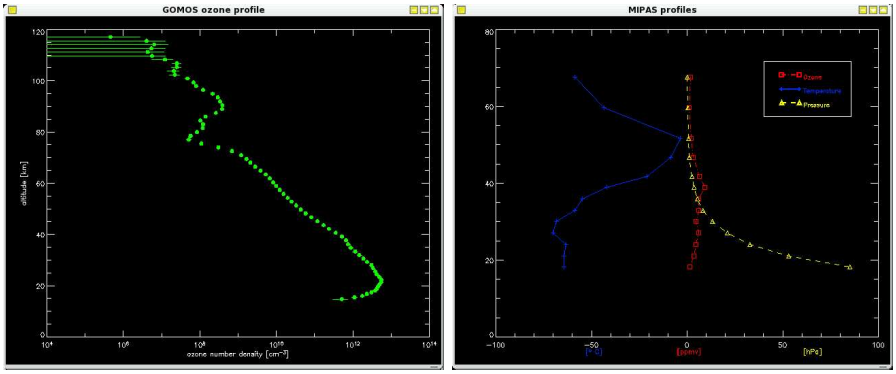


Figure 2.2: *Profiles of the selected data sets. Left panel:* GOMOS *ozone profile in number densities. Right panel:* MIPAS *ozone profile in mixing ratios, temperature profile in* $°C$ *and pressure in* hPa.

MIPAS data that is closer to the GOMOS profile than 200 km, we can run these commands:

```
IDL> distance = 200
IDL> codata = beatl2_find_colocated_data(mipasdata, gomosdata, $
                                         -1, distance, -1)
```

The negative values for the time and altitude limit indicate that these limits should be ignored in the search for co-located data. Doing so will yield a subset from the MIPAS data that is shown in red on the world map (Figure 2.1). Time can also be used as a constraint, as a first crude filter on the data, e.g., to select only data that were taken less than one second from the gomos profile.

Now we will concentrate on the selected data. The GOMOS ozone profile values are given in number densities, whereas MIPAS ozone values are given in mixing ratios, see Figure 2.2. Note that errorbars in a logarithmic plot (like the GOMOS ozone number densities plot) can be a bit deceiving; the errors may look small where they can be rather substantial. The MIPAS profile nicely shows the stratospheric ozone maximum at 40 kilometers, whereas the GOMOS profile shows the secondary ozone maximum in the mesosphere at about 90 kilometers altitude. To compare both ozone profiles, we need to convert the ozone mixing ratios of MIPAS into number densities (or vice versa the GOMOS data into mixing ratios).

The mixing ratio c_x of a gas x is the number of moles of x per mole of air, given here in parts per million volume [ppmv]. The mixing ratio is constant under temperature and pressure changes and therefore a robust measure of the atmospheric composition.

The number density n_x of a gas x is defined as the number of molecules of x per unit volume of air, given here in cm^{-3}. It is useful for calculating the atmospheric column of a optical active gas like ozone.

The number density and the mixing ratio of a gas are related by the number density of air n_a: $n_x = c_x n_a$. The number density of air can be determined

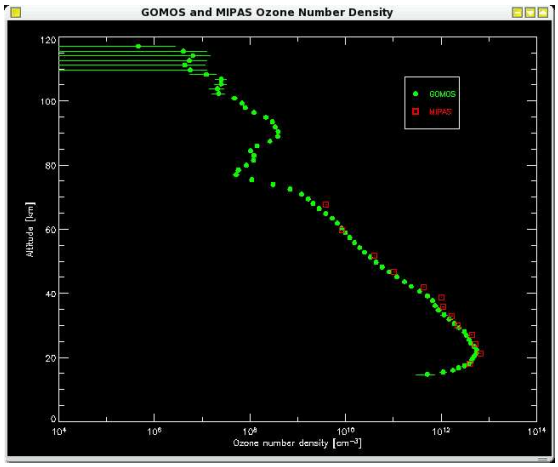


Figure 2.3: *Ozone profiles of* GOMOS *(green circles) and* MIPAS *(red squares) of the selected data as a function of altitude.*

with the ideal gas law,

$$PV = NRT, \tag{2.1}$$

where P and T are the pressure and temperature of a volume V of atmosphere, containing N number of molecules, respectively. R = 8.31 J mol^{-1} K^{-1} is the gas constant, and

$$n_a = \frac{A_v N}{V}, \tag{2.2}$$

where $A_v = 6.023 \cdot 10^{23}$ molecules·mol^{-1} is Avogadro's number. Combining all this we get

$$n_x = \frac{A_v P}{RT} c_x. \tag{2.3}$$

So in order to convert the ozone mixing ratios of MIPAS into number densities, we need the temperature and the pressure profiles. The pressure is given in the structure containing the ozone values. This can be checked by typing `HELP, codata, /STRUCTURE`

The temperature can be loaded with the same command as we used to load the ozone data, except that the filter will now be 'T' for temperature:

```
IDL> mipastemp = beatl2_ingest(filenames,'species=T,include_nan=1')
```

This is where the `include_nan` option comes in: without it only valid temperature data will be read and these might be different measurements than valid ozone measurements. With the `include_nan` option all data are read, so the records have definitely the same size and can be compared. The temperature profiles must of course be filtered in the same way as the ozone profiles. This will be done automatically and the temperature, pressure and ozone profiles will be shown in a plot, see the right panel of Figure 2.2.

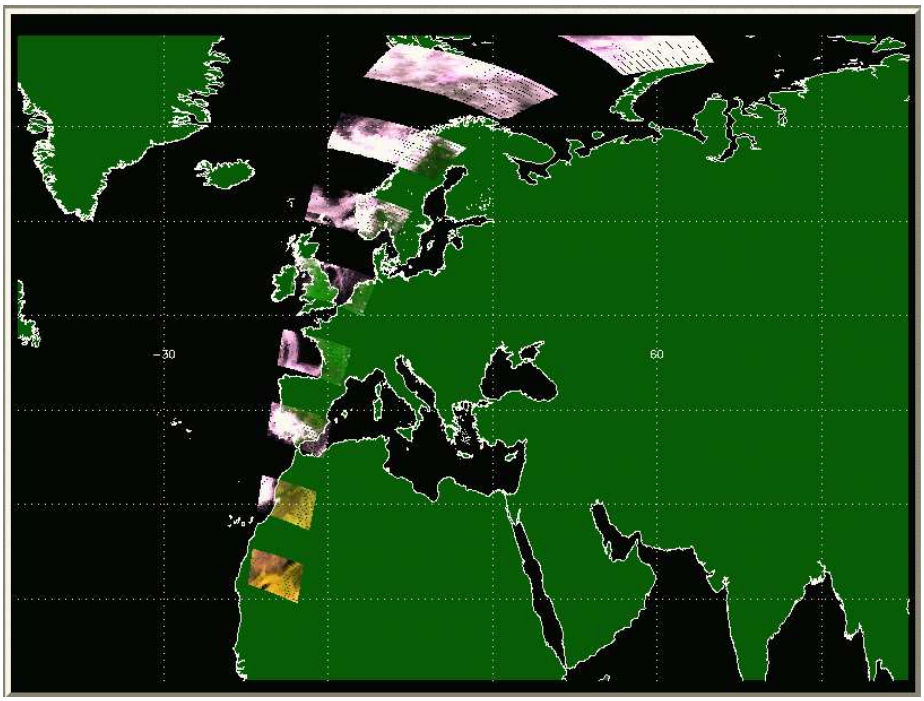
Using the pressure and temperature, the MIPAS ozone number density profile can be determined and compared to the GOMOS ozone number density, see Figure 2.3.

2.4 The BEAT routines

This program is an example of the use of the BEAT-II software. All datasets are loaded using simple commands and the returned data structures are intuitive and can be used directly. All data described in the BEAT-II data description can be loaded with no a priori knowledge of the data structures of the instruments. The only BEAT routines used in this program are

```
BEATL2_VERSION()
BEATL2_INGEST()
BEATL2_FIND_COLOCATED_DATA()
```

3 PMD imager



3.1 The problem

Polarisation Measurement Devices (PMDs) are used to determine the degree of atmospheric polarisation of the incoming light measured by SCIAMACHY. There are seven PMDs on-board SCIAMACHY, which measure the polarisation parameters in relatively broad spectral bands, see table [3.1]. Because of this low spectral resolution the spatial resolution of the PMDs can be higher than that of the normal radiance measurements of SCIAMACHY. The read-out frequency of the PMDs is 32 Hz, which yields a pixel size on the Earth of 7.5×30 km^2 in Nadir mode. This is four to eight times smaller than the normal pixel size of the Nadir mode of SCIAMACHY and therefore the PMDs can be used to make relatively high resolution true colour images of the Earth.

PMD	wavelength [nm]	(width) [nm]
1	344	(67)
2	487	(75)
3	661	(88)
4	853	(95)
5	1577	(137)
6	2322	(115)
7	854	(103)

Table [3.1]: SCIAMACHY *PMD frequencies and channel width.*

3.2 The work that needs to be done

To make true colour images of the Earth the following steps must be taken:

- The reflectance measured by the PMDs must be determined; the radiation measurements are given in binary units (BU), which do not have a physical unit, but the PMD measurements (in BU), the solar spectrum measurements (in BU) can be used to derive a reflectance. Also the geometry of each measurement is needed.
- A true colour image can be made with Red, Green and Blue; PMD-3 can be used for red ($\sim$ 675 nm) and PMD-2 for blue ($\sim$ 450 nm). The green colour ($\sim$ 525 nm) can be determined from a combination of these two channels, but we will take the PMD-4 signal as the green colour. This PMD measures in Infrared (IR) and vegetation is known to reflect strongly both the colour green and IR, so we will use IR as an artificial green colour.
- The colour pixels must be drawn on a world map; therefore the geolocations (corner coordinates) of the pixels must be read. Because the PMDs are read out more frequently than the normal measurements and no special PMD geolocations are provided, we need to determine the corner coordinates by interpolation.
- In order to make a nice plot, the backward scans must be removed.

3.3 Overview of the `IDL` program

The `IDL` program `pmd_tutorial.pro` (in the directory `pmd-imager`) will guide the user through the steps of the previous section. Here some background information will be given about the program.

To start the tutorial program, go to the `pmd-imager` directory and start `IDL` or `IDLDE`. From the `IDL` prompt type

```
IDL> pmd_imager
```

or in `IDLDE` load the file `pmd_imager.pro`, compile (`CTRL+F5`) and run (`F5`) it.

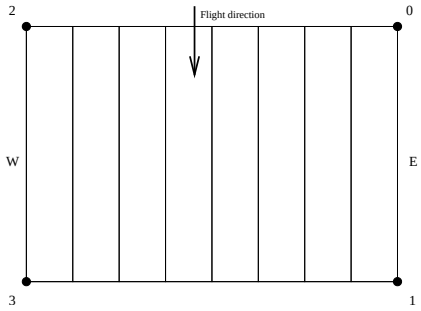


Figure 3.1: *Definition of the corner coordinates of a* SCIAMACHY *nadir pixel.*

The user is asked to load a SCIAMACHY file. This must be a level 1c file, containing at least PMD measurements. A sample file is provided, but any other SCIAMACHY level 1c file will do, provided it contains PMD measurements. From any SCIAMACHY level 1b file such a file can be created by calling the SciaL1C tool with the options for all calibrations (`-allcal`) and PMD measurements (`-pmd`). The sample file also contains normal nadir measurements from clusters 1-31, so it was created using[1]:

```
~> SciaL1C -allcal -pmd -type nadir -cluster 1,2,..,31 $SCIAfile
```

where `$SCIAfile` is the input level 1b file.

Instructions on what to do next is printed on screen (at the prompt in `IDL` or in the Log window of `IDLDE`). The program will continue each time after typing `.continue` (or just `.c`).

The first step is to identify the datasets that are in the file. This can be achieved using the `beat_datasetnames()` function. It lists the principal datasets present in the file. We are interested in the PMD measurements so we need the `nadir_pmd` dataset.

The `nadir_pmd` dataset (and the other datasets present) can be accessed with the `beat_fetch()` function. The data can be accessed for any state present in the file. We will process only one state for simplicity, this can be any state in the file indicated by the integer `state`. The user can change this integer to any number (note that `IDL` counts array indices starting at zero).

The measurements of all 7 PMDs are stored in a SCIAMACHY file as a one dimensional array per state, first all measurements of 1 PMD of one state, then all measurements of the second PMD, etc. This one dimensional array (type `help, pmds.int_pmd` to verify this) can be split up into a matrix with the measurements of the seven PMDs in columns, using

```
pmd = reform(pmd,7,npmd/7)
```

where `npmd` is the total number of PMD measurements. This is done automatically.

Next, the geolocations are extracted, using `beat_fetch()`. The longitudes and latitudes in `corner_coord` are used. This is a bit more work because

[1] all clusters must be given one by one, separated by comma's

of the way the geolocations are nested. BEAT handles these issues using `BEAT_DATAHANDLES`, see the BEAT documentation. The latitudes and longitudes are extracted using this piece of code:

```
FOR igeo = 0,ngeo-1 DO BEGIN
  geon = BEAT_FETCH(pmds.GEO[igeo])
  FOR i = 0,3 DO BEGIN
    lat[i,igeo] = geon.CORNER_COORD[i].LATITUDE
    lon[i,igeo] = geon.CORNER_COORD[i].LONGITUDE
  ENDFOR
ENDFOR
```

Because the PMDs are read out more frequently than the normal measurements, the geolocations of the intermediate PMD measurements must be determined by interpolation of the corner coordinates. This is done in the obvious way, as fractions of the corner coordinates 0 and 2 and the corner coordinates 1 and 3, respectively, see Figure 3.1. The interpolated coordinates are written to the arrays `lon_dev` and `lat_dev`.

The values of the PMD measurements determine the colour that must be used to fill the pixel defined by these coordinates. First the measurements, which are given in byte counts, must be transformed to reflectivities R, which is the fraction of the sunlight that is reflected by the atmosphere and the surface and detected by the sensor. It is defined as

$$R = \frac{\pi I}{\mu_0 E_0}, \tag{3.1}$$

where I is the radiance measured by the sensor, μ_0 is the cosine of the solar zenith angle (sza) and E_0 is the solar irradiance. The solar irradiance is a quantity that is measured once a day and can be found in the `sun_reference` dataset of the open file. The quantity that is needed here is the solar irradiance averaged over the (broad) PMD spectral region, this is the 7 values array `mean_pmd`. The sza can be found in the `geon` structure. It is an array of three values, corresponding to the east, middle and west points of each pixel. We will take the sza for the middle point of the pixel. Fetch these quantities by following the instructions.

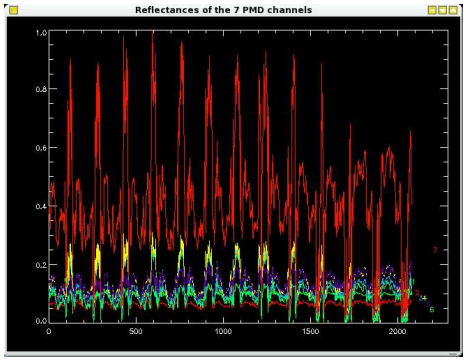


Figure 3.2: *Reflectances of the 7 PMD channels.*

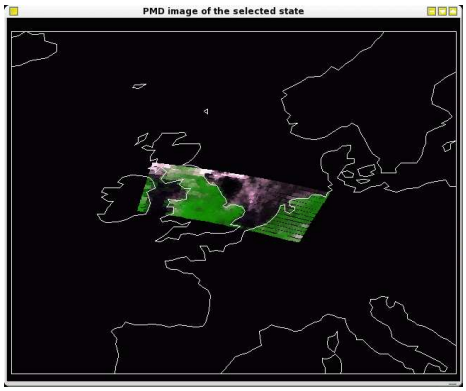


Figure 3.3: *Example of an RGB plot of a state over the Earth, based on PMD measurements. Black colours indicate ocean, white/pinkish colours indicate clouds, green indicates vegetation, yellow indicates deserts.*

Using all this the reflectivity measured by the PMDs can be determined. This is shown in a plot like Figure 3.2. All reflectances are between 0 and 1. Most of the signals will be rather close together, the signal of PMD-7 is different from the others, because it is configured in a different way. To use the PMD signals for an RGB picture, we need the PMDs closest to the colours red, green and blue, which are at approximately 675 nm, 525 nm and 450 nm, respectively. PMDs 2 and 3 can be used for blue and red, but the green colour is not represented by a different PMD channel, see table [3.1]. Therefore, we use PMD-4, which measures in the IR, because vegetation is known to reflect both the green colour as the IR strongly. The way to convert a three channel RGB colour to a one dimensional colour in `IDL` can be found in the `IDL` documentation: First the three signals must be converted to a byte array ranging from 0 to 255. Then the three channels can be combined using

```
IDL> rgb = red + 256 (green + (256 * blue)),
```

where `red`, `green` and `blue` are the byte arrays of the PMD signals.

Having done this we can plot the coloured pixels on a world map. This can be done using

```
IDL> map_set
IDL> for i=0,n_elements(data)-1 do $
     polyfill,lon_dev[*,i],lat_dev[*,i],color = data(i)
IDL> map_continents
```

This will be done automatically (type `.c`).

The plot will show you the selected state on a map of a piece of the Earth, like Figure 3.3. Clouds will be shown in a pinkish white colour, the sea is black, the land varies from green (if there is a lot of vegetation) to yellow (over deserts).

Typing `.c` once more will start a tool where all the states will be plotted in the same way as before. With this tool the way in which the geolocations are handled can be easily changed. Especially the backscan contains little information and obscures the plot, so it is desirable to remove it. The tool can also be

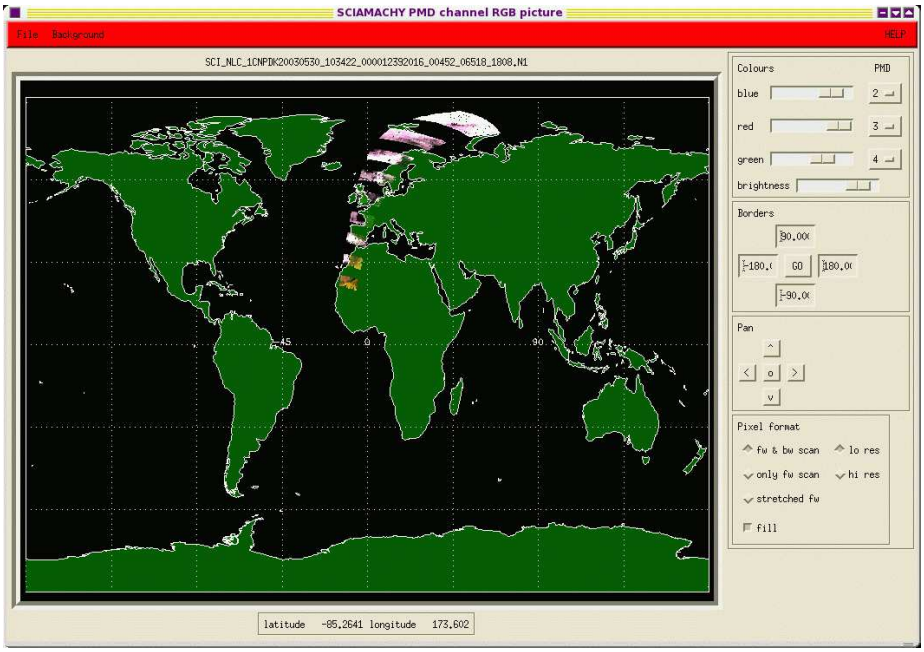


Figure 3.4: *Screen shot of the PMD imager tool.*

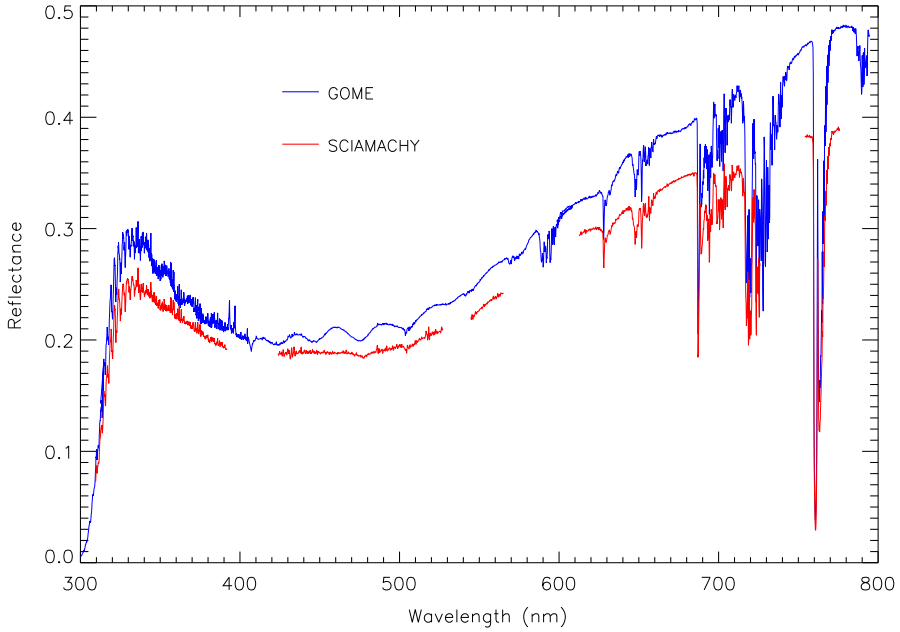
used to investigate the influence of using different PMD channels in the RGB picture. With the mouse the picture can be zoomed in and out. See the HELP function of the tool for more information.

3.4 The BEAT routines

This program is an example of a low level use of the BEAT-I software. All datasets are accessed directly, requiring an extensive knowledge of the SCIAMACHY data structures. The BEAT routines used in this program are

```
BEAT_UNLOAD
BEAT_OPEN()
BEAT_IS_ERROR()
BEAT_DATASETNAMES()
BEAT_FETCH()
BEAT_CLOSE()
```

4 SCIAMACHY and GOME reflectance comparison



4.1 The problem

SCIAMACHY and GOME are both spectrometers on-board space-borne platforms, measuring the Earthshine spectrum and Solar spectrum. SCIAMACHY was launched in May 2002 on-board ESA's ENVISAT, GOME was launched in April 1995 on-board the ERS-2 satellite. SCIAMACHY measures the Earthshine spectrum between 240 – 2380 nm with a resolution of 0.2 – 0.4 nm, while GOME measures from 237 – 794 nm, with a resolution of 0.2 – 0.4 nm.

Although well characterised in the laboratory, space-borne sensors must be calibrated in-flight using independent data to ensure the data quality. One opportunity to do this is to compare the reflectance spectrum of GOME with that of SCIAMACHY as both sensors fly in the same orbit, with GOME following SCIAMACHY by half an hour.

4.2 The work that needs to be done

To compare SCIAMACHY and GOME reflectance the following steps must be taken:

- A place on Earth must be selected where both GOME and SCIAMACHY measure in nadir mode (SCIAMACHY switches between nadir and limb measurements) and this scene is preferably homogeneous (i.e. the reflectance of the surface is constant over the area). GOME and SCIAMACHY spectra between 240 and 800 nm must be read and converted to reflectances.
- GOME pixels are about 320×40 km^2, while SCIAMACHY pixels are about 60×30 km^2. We will look at a rectangular spot on the Earth of 5 co-added GOME pixels and this area is filled with SCIAMACHY pixels which must also be co-added.
- The co-added reflectances can then be compared.

4.3 Overview of the IDL program

The IDL program `sciamachy_gome.pro` (in the `scia-gome` directory) will guide the user through the steps of the previous chapter.

To start the tutorial program, go to the `scia-gome` directory and start IDL or IDLDE. From the IDL prompt type

```
IDL> sciamachy_gome
```

or in IDLDE load the `sciamachy_gome.pro`, compile (`CTRL+F5`) and run (`F5`) it.

First the data files must be read. Two sample files are provided, a GOME file of orbit 19227 and a SCIAMACHY file of orbit 2509.

The GOME file (`GOME_*.el1`) is a simple ASCII file containing the Solar Spectrum of 16 Oct 2001 12:17:58 and the Earth shine Spectrum of 23 Aug 2002 starting at 09:38:28. It was made with the standard extraction software provided by ESA, using the call

```
~> gdp01_ex -r 32.6 4.0 26.9 35.0 $GOMEIN $ASCOUT
```

where `$GOMEIN` and `$ASCOUT` are the binary input and ASCII output files, respectively. The time difference between the Solar and Earth shine spectra is caused by a pointing problem of GOME: the exact position of the sun could not be determined anymore after October 2001. This causes a problem in the determination of the reflectance, because the Earth shine spectrum detectors are degrading since 1998. Usually a GOME file covers a whole orbit, but the sample file was stripped to reduce the file size, so only the necessary information is retained.

The SCIAMACHY file sample was described in section 3.3.

The geolocation First we will read the GOME data. Instructions on what to do next is printed on screen (at the prompt in IDL or in the Log window of IDLDE) screen. The program will continue each time after typing `.continue` (or just `.c`).

After opening the GOME file and accessing the data, the structure named `GOME_PRODUCT` can be examined to see what is in the file. The geolocations can be

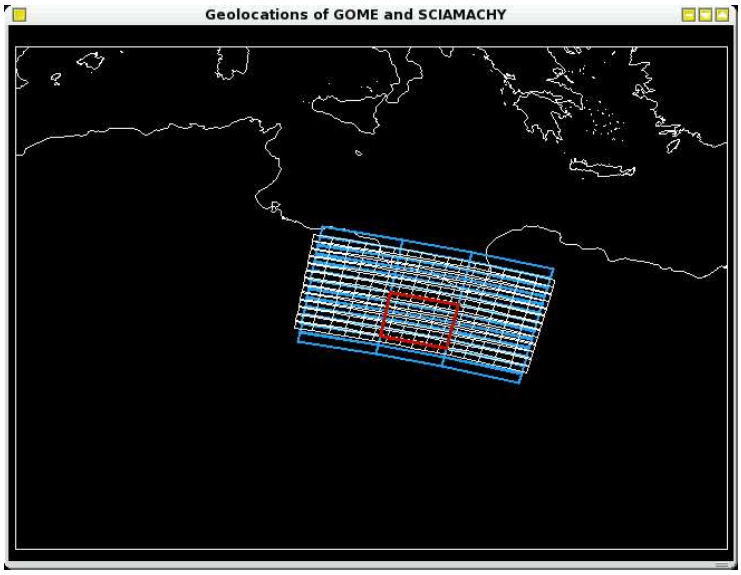


Figure 4.1: *The GOME pixels (blue), SCIAMACHY pixels (white) and the rectangle (red) defining the area of 5 adjacent GOME pixels.*

retrieved with the `GET_GOME_GEO.pro` procedure, using `BEAT_FETCH()` to access the data. This simple procedure fetches the geolocation data from the structure format and puts it into `IDL` arrays (`GOME_LONGITUDE, GOME_LATITUDE`). As an extra the pixel type (East, Nadir, West or Backscan, which is given in the GOME product) is also stored in the array `GOME_PIXELTYPE`.

After continuing the program will show a plot of the location of the GOME pixels on the Earth, similar to the blue lines in Figure 4.1.

Next, the SCIAMACHY file is opened and the data accessed. Follow the directions on screen. Note that although the GOME file is a simple ASCII file and the SCIAMACHY file has its own binary structure, both can be accessed using the same BEAT commands. The structure `SCIA_PRODUCT` can again be examined to see what is in the file (Type `HELP scia_product, /STRUCTURE`).

For the extraction of the SCIAMACHY geolocations there are several options, because of the internal structure of the SCIAMACHY data format, which reflects the measuring strategy. SCIAMACHY measures (almost) the entire spectrum between 240 and 2400 nm, but not all wavelengths are measured with the same accuracy. Depending on the channel and the place on Earth, measurements are co-added with an integration time (IT) of 0.125, 0.25, 0.5, 1.0, 5.0 or 10.0 s. The reason is to ensure high signal-to-noise ratio when the amount of radiation is low and/or to reduce the data rate over less interesting areas. A smaller IT results in smaller pixels and more detailed information. Usually between 60°N and 60°S the smallest integration time is 0.25 s except for certain parts of the spectrum and this is the case here, so we will continue with an IT of 0.25 s.

We must distinguish between forward scans and backward scans: Using `BEAT_FETCH(scia_product.STATES[5])` information on the measuring strategy can be retrieved (State 6 is the Sahara state that we are interested in, remember

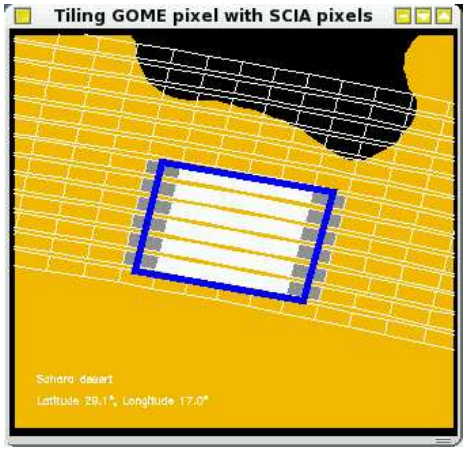


Figure 4.2: *Coverage fraction of the* SCIAMACHY *pixels. Open pixels are completely outside the selected rectangle, white pixels are completely inside the selected rectangle and shaded pixels are partly within the selected rectangle. The shading indicates the amount of coverage of the pixels, which will be used as a weighing function.*

that IDL starts to count at 0). The `STATE.DUR_SCAN_PHASE` gives the duration of measurement of the current state (65 s in this case). With an integration time of 0.25 s this means that `DUR_SCAN_PHASE/IT` number of measurements (260 in this case) have been performed. `STATE.NUM_DSR` gives the number of data set records (13 in this case), which is the number of scans in this state (forward + backward). So one scan contains 260/13 = 20 geolocations, of which the first 4/5th belong to the forward scan. This ratio is always constant (1/5th to East, 1/5 to Center-East, 1/5th to Center-West, 1/5th to West and 1/5th to backscan).

To extract the geolocation of SCIAMACHY the procedure `GET_SCIA_GEO.pro` is available. It will extract geolocations from the structure format into simple arrays. The state number, IT and ground pixel number are given as arguments to this procedure. Continuing the program will show the SCIAMACHY in the same plot, like the white pixels in Figure 4.1.

The scene that we will concentrate on here is a rectangular part of the Saharan desert with a more or less homogeneous surface albedo and which was cloud free on 23 August 2002. The rectangle is given in Figure 4.1 in red.

The spectra The next step is to retrieve the spectra of the pixels within the selected area and co-add them in the right way. For this the coverage of the SCIAMACHY pixels must be determined. This is done by determining for each SCIAMACHY pixel the area that is located within the rectangle. A plot like Figure 4.2 shows the coverage by shade of the each pixel. White pixels lie completely within the selected area, open pixels are completely outside the rectangle, and shaded pixels lie partly within the rectangle. The shading indicates coverage fraction. This coverage fraction can be used as a weighing factor during the co-adding of the reflectances of the pixels.

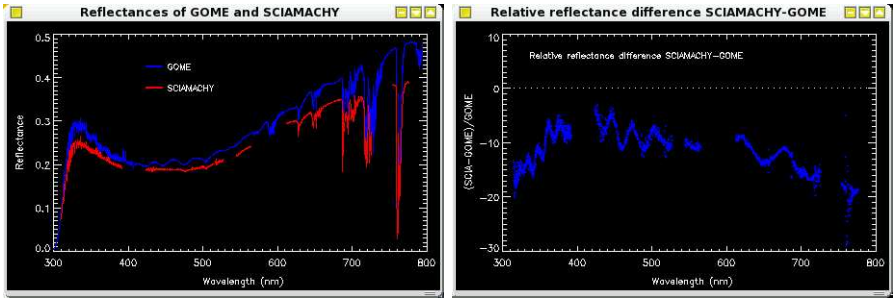


Figure 4.3: *Left panel: Reflectance of the 5 co-added* GOME *pixels (blue) and the reflectance of the co-added* SCIAMACHY *pixels lying (partly) within the selected rectangle (red). Right panel: Relative difference of the* GOME *and* SCIAMACHY *reflectance.*

There is now enough information to retrieve the spectra of the selected GOME pixels and the SCIAMACHY pixels lying withing the selected area. The spectra will be loaded automatically. The reflectance is defined in equation (3.1). We need the measurements from 4×1024 detector pixels for both GOME and SCIAMACHY, divide them by the solar measurements of the same detector pixels and the solar zenith angles. This is done in `READ_GOME_PIXEL.pro`. This routine can be called for each GOME pixel with arguments the BEAT product id of the open GOME file (`gome_product`), the groundpixel-number and a name for the output. It returns a structure named `spectrum`, containing pixel information, reflectances and wavelengths and geolocations. So the call for each pixel `i` is

```
IDL>  READ_GOME_PIXEL, gome_product, gome_pixels[i], gspectrum
```

Browsing the procedure, the user will find again that the same BEAT routine (`BEAT_FETCH()`) is used to extract all the information.

The SCIAMACHY data is retrieved using a call to `BIN_SCIA_PIXELS.pro`. It retrieves the spectrum information of all or several pixels in a state with a certain integration time and co-adds them. It has as arguments the BEAT product id of the open SCIAMACHY file (`scia_product`), the state number, the requested integration time, an array of the ground pixels indices that must be co-added, the array of the weights of the pixels and a name for the output. So the call for each state is

```
IDL>  BIN_SCIA_PIXELS, scia_product, stateID, IT, pixelNr,    $
                       coverage, sspectrum
```

It also returns a structure also named `spectrum`, which contains all necessary information. Again one can find that this procedure also retrieves all information using the BEAT routine `BEAT_FETCH()`.

Having read the spectral information, we can plot and compare the data from GOME and SCIAMACHY, see figure 4.3. The SCIAMACHY reflectance is $10 - 20\%$ lower than the GOME reflectance. This is a well-known offset also found by other comparisons with known reflectance spectra and the GOME reflectance has been used to correct the SCIAMACHY reflectances. There are some gaps in the SCIAMACHY plot, because we have only retrieved information of pixels with

an integration time of 0.25 s. In this state there are some spectral regions in which a longer integration time than 0.25 s was used. This is mainly in the overlap regions, i.e. the regions at channel boundaries.

4.4 The BEAT routines

This program is an example of the use of the BEAT-I software for different files formats. Although GOME and SCIAMACHY file formats are very different internally, the same BEAT routines can be used for the same actions on both file types. The BEAT routines used in this program are

```
BEAT_VERSION()
BEAT_UNLOAD
BEAT_OPEN()
BEAT_IS_ERROR()
BEAT_FETCH()
BEAT_TIME_TO_STRING()
BEAT_CLOSE()
```