Stereo-Photogrammetric ATSR Data Analysis with BEAM

Ludwig M Brinckmann
lmb2@mssl.ucl.ac.uk

March 20, 2008

Abstract – BEAM is software toolkit for earth observation data visualisation and analysis. This document is a user guide to the mssl-stereomatcher extension module to BEAM for the stereo-photogrammetric analysis of ATSR data.

The module supports two application cases: the retrieval of geometric cloud-top height and the assessment of the co-registration error in ATSR data. The first utilises the Mannstein camera model to translate parallax into height, while the second uses clear views of land as ground control points to determine the presence and size of any shift between the nadir and the forward view.

This document describes the installation, usage, both interactive and in batch processing, and the internal implementation of the module. The scientific background to the functionality is covered in Muller et al. (2007), Denis et al. (2007) and Moroney et al. (2002).

This document first covers software installation before giving an introduction to its use interactively via the BEAM graphical user interface (GUI) and non-interactively via the BEAM graph processing framework (GPF).

1 Installation

This section describes the installation of the MSSL stereo-matcher module in BEAM. While the installation of BEAM is outside the scope of this document, it is straightforward. The BEAM installation requires a recent Java installation. After installation of BEAM *first* verify that BEAM operates correctly. Once Java and BEAM are installed the process of installing the MSSL stereo-matcher is identical in Unix and Windows environments.

1.1 Software Installation

The stereo matching module requires version 4.2 of the BEAM toolkit, which can be downloaded from `http://www.brockmann-consult.de/beam-wiki/display/BEAM/Latest+Development+Snapshots`. Version 0.8 of the stereo matching module is available for download at `ftp://ftp.mssl.ucl.ac.uk/pub/imaging/lmb2/mssl-stereomatcher-0.8.jar`. After download, the `mssl-stereomatcher-0.8.jar` must be copied into the `modules` directory in the BEAM installation. If upgrading from a previous version of the module, the previous `.jar` file must be removed.

BEAM can then be started normally and a number of new menu items should appear in the `tools` menu at the top of the BEAM Visat GUI. Fig. 1 shows the additional tools, conventionally prefixed by `MSSL`. The BEAM Graph Processing Tool GPT should also list new operators. The output from the GPT tool should look similar to this:

```
>gpt.sh
...
Operators:
```

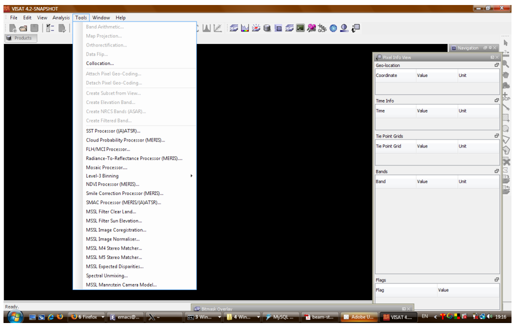

VISAT 4.2-SNAPSHOT
File
Edit
View
Analysis
Tools
Window
Help
Products
Band Arithmetic...
Map Projection...
Orthorectification...
Data Flip...
Collocation...
Attach Pixel Geo-Coding...
Detach Pixel Geo-Coding...
Create Subset from View...
Create Elevation Band...
Create NRCS Bands (ASAR)...
Create Filtered Band...
SST Processor ((A)ATSR)...
Cloud Probability Processor (MERIS)...
FLH/MCI Processor...
Radiance-To-Reflectance Processor (MERIS)....
Mosaic Processor....
Level-3 Binning
NDVI Processor (MERIS)...
Smile Correction Processor (MERIS)...
SMAC Processor (MERIS/(A)ATSR)...
MSSL Filter Clear Land...
MSSL Filter Sun Elevation...
MSSL Image Coregistration...
MSSL Image Normaliser...
MSSL M4 Stereo Matcher...
MSSL M5 Stereo Matcher...
MSSL Expected Disparities...
Spectral Unmixing...
MSSL Mannstein Camera Model...
Navigation
Pixel Info View
Geo-location
Coordinate
Value
Unit
Time Info
Time
Value
Unit
Tie Point Grids
Tie Point Grid
Value
Unit
Bands
Band
Value
Unit
Flags
Flag
Value
Ready.
Bitmask Overlay

Figure 1: BEAM showing additional tools for stereo-matching in the tools drop down menu.

```
Normaliser                Normalises an input image
Read                      Reads a product from disk.
ImageCoregistration       Determines nadir/forward coregistration by determining sh
ExpectedDisparities       Computes expected disparities from elevation
Write                     Writes a product to disk.
Unmix                     Performs a linear spectral unmixing.
PassThrough               Sets target product to source product.
MannsteinCameraModel      Computes geometric height from parallax using Mannstein C
Collocate                 Collocates two products based on their geo-codings.
SunElevationDataFilter    Filters input data based on sun elevation
M4StereoMatcher           Stereo matches using M4 Algorithm
ClearLandFilter           Filters input to be both clear and over land
M5StereoMatcher           Stereo matches using M5 Algorithm
```

Caveat: Versions of BEAM prior to version 4.2 will not work and as Java is a compiled, statically typed language, it is likely that future version of BEAM will not be able to load the stereo matching module without recompilation. The directions for recompilation can be found in section 4.4.

1.2 Data

The stereo-matcher works on scenes of ATSR data in the Envisat format of a minimum size of 512*512 pixels. A test scene is available for download at `http://141.4.215.13/data/products/ATS_TOA_1CNPDK20030504_111142_000000772016_00080_06146_0157.zip`. This scene is used throughout this document as illustration as it illustrates the working of the stereo-matcher while being suitable to interactive processing due to its small size. Larger data sets are available from either the NERC Earth Observation Data Centre NEODC `http://www.neodc.rl.ac.uk` or ESA.

It is also possible to to produce a spatially reduced data set of ATSR data using BEAM. Such a data set can be stored in the BEAM-DIMAP format and can be used just like an original ATSR data product.

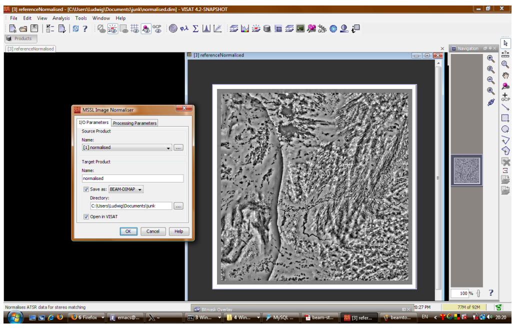
2 Architecture

BEAM's native functionality can be extended through *modules*. A module comprise a number of functions that at the user level are known as *operators*. Operators can have any number of *source products* and produce a *target product*. The functionality of an operator can be influenced by setting *processing parameters*.

The functionality of the MSSL stereo-matcher is broken down into several operators that must be chained together to either retrieve geometric cloud-top height or to determine the co-registration shift in ATSR data. The chain begins with an ATSR data file to produce intermediate target products that serve as source products for the next step.

This section briefly introduces the available operators:

MSSL Image Normaliser: The image normaliser operator implements the normalisation function described in Muller et al. (2007). It takes two input bands, which can be set as processing parameters, and maps these into a three target bands each: a normalised view of the data, a regional standard deviation and a regional mean. For convenience the input bands are copied into the target product. An example use is illustrated in fig. 2.


[3] referenceNormalised - [C:\Users\Ludwig\Documents\junk\normalised.dim] - VISAT 4.2-SNAPSHOT

MSSL Image Normaliser

I/O Parameters

Source Product
Name: [1] normalised

Target Product
Name: normalised
[✓] Save as: BEAM-DIMAP
Directory: C:\Users\Ludwig\Documents\junk
[✓] Open in VISAT

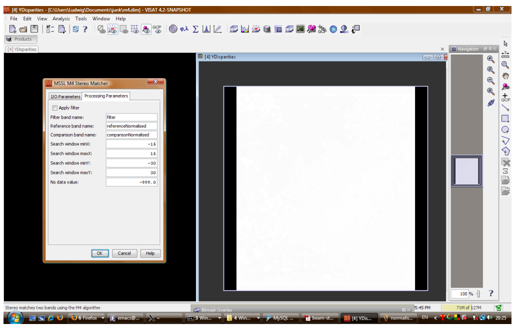
Figure 2: BEAM showing MSSL Image Normaliser operator and result image.

MSSL Filter Sun Elevation: This operator implements a spatial filter to reduce the dataset that will be processed by the stereo-matcher based on the sun-elevation data contained in the ATSR file. If it is desired to process only day-light data, the sun elevation processing parameter could be set to 10.0. This will produce a target product containing a band with the name `filter` where only values with a higher sun elevation are set to 1. The stereo matching module will ignore data where the value is set to any other value. The use of this operator is not required.

MSSL M4 Stereo Matcher: The M4 stereo matcher implements the fast M4 stereo matcher based on phase correlation shift (Muller et al. 2007). It takes as source products the output of the image normaliser and optionally a filter. A filter source product must always be set (a limitation in the BEAM GUI), but is only activated if the `Apply filter` box is ticked. Other processing parameters include the size of the search window and the value for no data. The target product contains three output bands: YDisparities, containing the disparity in the along-track direction, XDisparities, containing the disparity in the across-track direction, and the match quality. The stereo-matching is the most time consuming task and processing an entire orbit can take as long as one hour on a current personal computer. An example use is illustrated in fig. 3.

MSSL M5 Stereo Matcher: The M5 stereo matcher implements a stereo-matcher very similar to M4, but with an exhaustive search within the search window. The same processing parameters as with M4 apply.

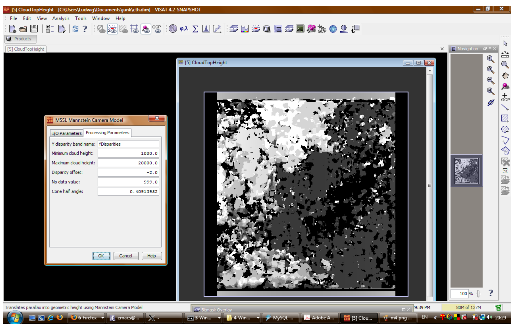
MSSL Mannstein Camera Model: This operator translates along-track disparities into geometric cloud-top height using the Mannstein camera model (Denis et al. 2007). Processing parameters are minimum and maximum cloud-top height, a disparity offset to compensate for a shift in co-registration, and the instrument half-cone angle. An example use is illustrated in fig. 4.



Apply filter: [ ]
Filter band name: filter
Reference band name: referenceNormalised
Comparison band name: comparisonNormalised
Search window minX: -15
Search window maxX: 15
Search window minY: -30
Search window maxY: 30
No data value: -999.0

Stereo matches two bands using the M4 algorithm

Figure 3: BEAM showing MSSL M4 Stereo Matcher operator and result image.

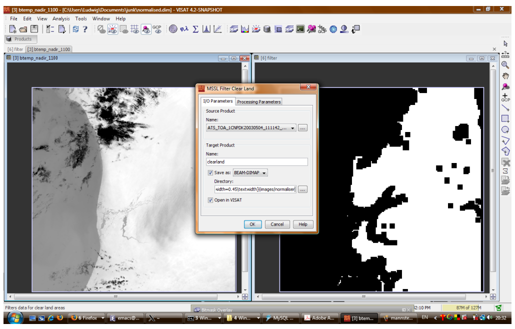

[5] CloudTopHeight - [C:\Users\Ludwig\Documents\junk\cth.dim] - VISAT 4.2-SNAPSHOT
File
Edit
View
Analysis
Tools
Window
Help
Products
MSSL Mannstein Camera Model
I/O Parameters
Processing Parameters
OK
Cancel
Help

Y disparity band name: YDisparities
Minimum cloud height: 1000.0
Maximum cloud height: 20000.0
Disparity offset: -2.0
No data value: -999.0
Cone half angle: 0.40913952

Translates parallax into geometric height using Mannstein Camera Model

Figure 4: BEAM showing MSSL Mannstein Camera Model operator and result image.

MSSL Filter Clear Land: This operator assists in the detection of the co-registration shift by masking out all areas that are either over sea or cloudy. As the stereo-matching process is affected by clouds in the vicinity of the pixel analysed the processing parameter `cloud radius` gives the radius of cloud-free pixels required to classify a pixel as clear. An example use is illustrated in fig. 5.



Source Product
Name: ATS_TOA_1CNPDK20030504_111142_...
Target Product
Name: clearland
[✓] Save as: BEAM-DIMAP
Directory: width=0.45\textwidth]{images/normaliser
[✓] Open in VISAT

Figure 5: BEAM showing MSSL Clear Land operator and result image.

MSSL Expected Disparities: When determining the co-registration shift expected disparities or parallax arising from terrain must be taken into account. This operator computes the expected parallax based on the assumption that every 800m of elevation will result in a one pixel disparity.

MSSL Image Coregistration: This operator takes three source products: disparities resulting from one of the stereo matchers (M4 or M5), a viewfilter (e.g. clear land filter) and expected disparities and computes an output band of elevation corrected disparities, which should give an indication of the image shift present between nadir and forward view.

3 Applications

After the brief introduction of the operators contained in the module, this section describes the workflow for cloud-top height and co-registration shift retrieval.

While the previous brief introduction to the operators gave only instructions for the interactive use, here also the use within the BEAM Graph Processing Framework GPF is described. The GPF is built upon the concept of a workflow of data through a directed, acyclic network of processing nodes driven by a desired target product. The BEAM GPT tool reads a graph in XML format which contains descriptions of nodes, implemented by the operators described above. The nodes are chained together by their respective source and target products.

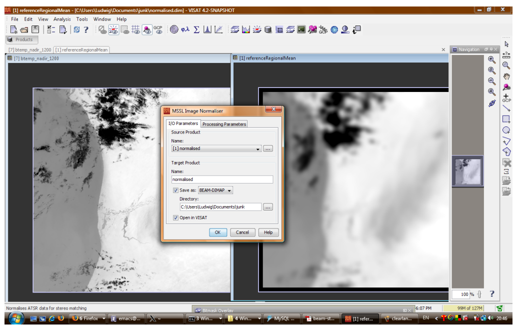
3.1 Photogrammetric Cloud Top Height Retrieval

The complete chain for the retrieval of cloud-top height from ATSR data is combinded of four operator:

- Image Normalisation
- Creation of a spatial filter (optional)
- Stereo Matching using M4 or M5
- Translation into geometric height using the Mannstein camera model

3.1.1 Interactive Use

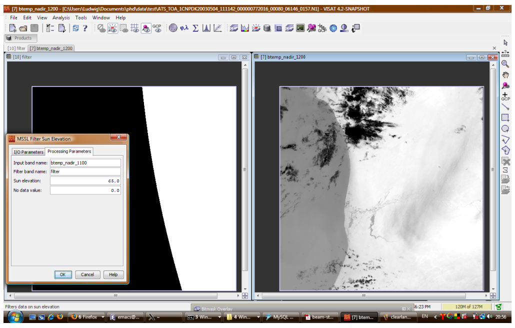
- First an ATSR source product must be normalised with the `MSSL Image Normaliser`. Open the tool from the `Tools` menu and select an ATSR input file. By default the band processed are the 12μm band, but this can be changed in the processing parameters panel. In the target product will be eight bands that will serve as input data for the stereo-matcher. Fig. 6 shows the regional mean.



Source Product
Name: [1] normalised
Target Product
Name: normalised
[✓] Save as: BEAM-DIMAP
Directory: C:\Users\Ludwig\Documents\junk
[✓] Open in VISAT

Figure 6: BEAM showing MSSL Image Normaliser operator and result image.

- The optionally a filter can be created. In this example we use for exposition only a filter value of 65 to mask out some areas. The result is shown in fig. 7.


MSSL Filter Sun Elevation

I/O Parameters | Processing Parameters

Input band name: btemp_nadir_1100

Filter band name: filter

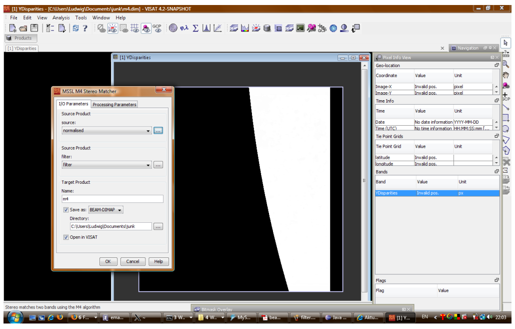
Sun elevation: 65.0

No data value: 0.0

Filters data on sun elevation

Figure 7: BEAM showing MSSL Sun Elevation operator and result image.

- The next step applies the stereo matcher to the scene. To apply the filter the `Apply filter` box must be ticked. The result, with the filter applied, is shown in fig. 8.


YDisparities - [C:\Users\Ludwig\Documents\junk\m4.dim] - VISAT 4.2-SNAPSHOT

MSSL M4 Stereo Matcher

I/O Parameters | Processing Parameters

Source Product

source: normalised

Source Product

filter: filter

Target Product

Name: m4

[✓] Save as: BEAM-DIMAP

Directory: C:\Users\Ludwig\Documents\junk

[✓] Open in VISAT

Stereo matches two bands using the M4 algorithm

Figure 8: BEAM showing MSSL M4 Stereo Matcher operator with filter and result image.

- The data resulting from the stereo-matcher and the filter is then processed into cloud-top height with the MSSL Mannstein Camera Model.The result, with the filter applied, is shown in fig. 9.

3.1.2 Batch Use

Batch use requires an XML graph description for the processing chain, such as in this example:

```
<graph>
   <id>StereoMatcher</id>
   <node>
     <id>SunElevationDataFilter</id>
     <operator>SunElevationDataFilter</operator>
     <sources>
       <atsrToaL1b>${atsrToaL1b}</atsrToaL1b>
     </sources>
     <parameters>
       <filterBandName>filter</filterBandName>
       <noDataValue>0</noDataValue>
       <inputBandName>reflec_nadir_1100</inputBandName>
       <sunElevation>10.0</sunElevation>
     </parameters>
```

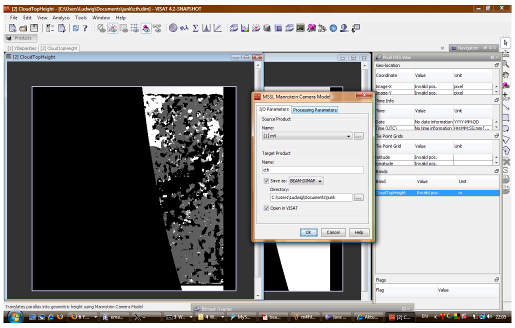


Figure 9: BEAM showing MSSL Mannstein Camera Model operator with filter and result image.

```
</node>
<node>
  <id>Normalisation</id>
  <operator>Normaliser</operator>
  <sources>
    <atsrToaL1b>${atsrToaL1b}</atsrToaL1b>
  </sources>
  <parameters>
    <referenceBandName>btemp_nadir_1100</referenceBandName>
    <comparisonBandName>btemp_fward_1100</comparisonBandName>
  </parameters>
</node>
<node>
  <id>StereoMatching</id>
  <operator>M4StereoMatcher</operator>
  <sources>
    <source>Normalisation</source>
   <filter>SunElevationDataFilter</filter>
  </sources>
  <parameters>
   <searchWindowMaxX>15</searchWindowMaxX>
    <searchWindowMaxY>30</searchWindowMaxY>
    <referenceBandName>referenceNormalised</referenceBandName>
    <noDataValue>-999.0</noDataValue>
    <searchWindowMinY>-30</searchWindowMinY>
    <searchWindowMinX>-15</searchWindowMinX>
    <comparisonBandName>comparisonNormalised</comparisonBandName>
    <applyFilter>true</applyFilter>
    <filterBandName>filter</filterBandName>
  </parameters>
</node>
<node>
  <id>MannsteinModel</id>
  <operator>MannsteinCameraModel</operator>
  <sources>
    <source>StereoMatching</source>
  </sources>
  <parameters>
```

```
        <yDisparityBandName>YDisparities</yDisparityBandName>
        <disparityOffset>-2</disparityOffset>
        <maximumCloudHeight>20000.0</maximumCloudHeight>
        <noDataValue>999.0</noDataValue>
        <minimumCloudHeight>1000.0</minimumCloudHeight>
      </parameters>
    </node>
  </graph>
```

Here the node with `id` `Mannstein Model` will force the computation of its source product, which is the output from node `StereoMatching`, which in turn will require the computation of the nodes `Normalisation` and `SunElevationDataFilter`. The processing parameters for each node are given and can easily be adapted.

The processing chain can then simply be executed by a call to the BEAM gpt command with an ATSR data source file. The `-t` parameter specifies the name of the output file, the `-SatsrToaL1b` parameter the ATSR input file, the `-f` parameter the output file format.

```
gpt.sh cthchain.xml -t output.dim \
-SatsrToaL1b=ATS_TOA_1POLRA20030401_024113_000062202015_00103_05668_0591.N1 \
-f BEAM-DIMAP
```

3.2 Co-Registration Shift Assessment

The assessment of the co-registration shift of ATSR data follows the ideas laid out in Moroney et al. (2002) for MISR. The workflow is as follows:

- Image Normalisation
- Creation of a spatial filter (optional)
- Stereo Matching using M4 or M5
- Computation of expected disparities
- Creation of a clear land mask
- Computation of residual registration shift

3.2.1 Interactive Use

The initial steps are identical to the computation of cloud-top height and are not repeated here.

- Expected disparities are computed from elevation data contained in the ATSR product. A sample result, this time showing the Red Sea[1], is shown in fig. **??**.
- A clear land mask is computed by the Clear Land Mask operator.
- The results from the previous operations are then combined to calculate the residual disparity which should give an indication of the co-registration shift using the operator `MSSL Image Coregistration` . This is illustrated in fig. 11.

[1]The usual test-scene does not have enough variation in elevation to demonstrate the operator.

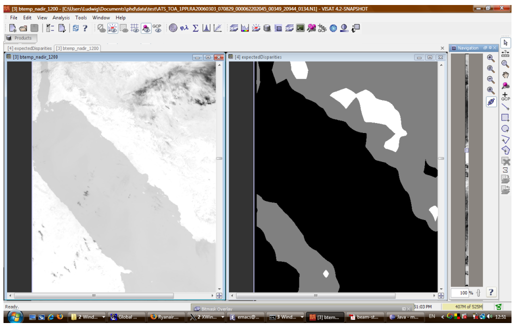


Figure 10: BEAM showing MSSL Expected Disparities result image.

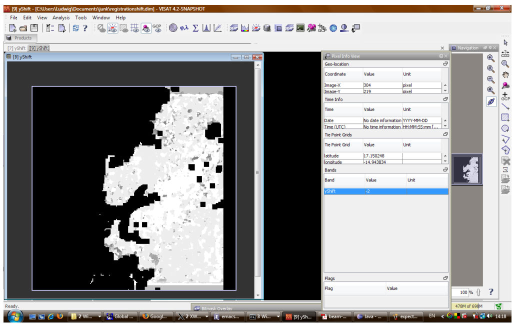


Figure 11: BEAM showing MSSL Image Coregistration operator and a result image, with constrast stretched.

3.2.2 Batch Use

As for the cloud-top height retrieval the image registration can also be computed in batch mode with the GPT command. The graph description file will need to be similar to this:

```
<graph>
  <id>ImageCoregistration</id>
  <node>
    <id>SunElevationDataFilter</id>
    <operator>SunElevationDataFilter</operator>
    <sources>
      <atsrToaL1b>${atsrToaL1b}</atsrToaL1b>
    </sources>
    <parameters>
      <filterBandName>filter</filterBandName>
      <noDataValue>0</noDataValue>
      <inputBandName>btemp_nadir_1100</inputBandName>
      <sunElevation>10.0</sunElevation>
    </parameters>
  </node>
  <node>
    <id>Normalisation</id>
    <operator>Normaliser</operator>
    <sources>
      <atsrToaL1b>${atsrToaL1b}</atsrToaL1b>
    </sources>
    <parameters>
      <referenceBandName>btemp_nadir_1100</referenceBandName>
      <comparisonBandName>btemp_fward_1100</comparisonBandName>
    </parameters>
  </node>
  <node>
    <id>StereoMatching</id>
    <operator>M5StereoMatcher</operator>
    <sources>
      <source>Normalisation</source>
     <filter>SunElevationDataFilter</filter>
    </sources>
    <parameters>
      <searchWindowMaxX>30</searchWindowMaxX>
      <searchWindowMaxY>30</searchWindowMaxY>
      <referenceBandName>referenceNormalised</referenceBandName>
      <noDataValue>-999.0</noDataValue>
      <searchWindowMinY>-20</searchWindowMinY>
      <searchWindowMinX>-20</searchWindowMinX>
      <comparisonBandName>comparisonNormalised</comparisonBandName>
      <applyFilter>false</applyFilter>
      <filterBandName>filter</filterBandName>
    </parameters>
  </node>
  <node>
    <id>ExpectedDisparities</id>
    <operator>ExpectedDisparities</operator>
    <sources>
      <atsrToaL1b>${atsrToaL1b}</atsrToaL1b>
    </sources>
    <parameters>
      <noDataValue>-999</noDataValue>
      <outputBandName>expectedDisparities</outputBandName>
    </parameters>
  </node>
  <node>
    <id>ClearLandFilter</id>
    <operator>ClearLandFilter</operator>
    <sources>
      <atsrToaL1b>${atsrToaL1b}</atsrToaL1b>
    </sources>
    <parameters>
      <noDataValue>0</noDataValue>
      <filterBandName>filter</filterBandName>
      <cloudyRadius>5</cloudyRadius>
    </parameters>
  </node>
  <node>
    <id>ImageCoregistration</id>
    <operator>ImageCoregistration</operator>
    <sources>
      <viewFilter>ClearLandFilter</viewFilter>
      <disparities>StereoMatching</disparities>
      <expectedDisparities>ExpectedDisparities</expectedDisparities>
```

```
      </sources>
      <parameters>
        <yShiftBandName>yShift</yShiftBandName>
        <noDataValue>-999</noDataValue>
        <yDispBandName>YDisparities</yDispBandName>
        <xDispBandName>XDisparities</xDispBandName>
        <xShiftBandName>xShift</xShiftBandName>
        <expectedDispBandName>expectedDisparities</expectedDispBandName>
        <viewFilterBandName>filter</viewFilterBandName>
      </parameters>
    </node>
  </graph>
```

The batch processing is called with parameters as before:

```
gpt.sh registrationchain.xml -t output.dim \
-SatsrToaL1b=ATS_TOA_1POLRA20030401_024113_000062202015_00103_05668_0591.N1 \
-f BEAM-DIMAP
```

3.3 Creating Climatologies

BEAM provides an L3 binning processor for the creation of climatologies based on sinosoidal projection. The L3 binning processor can be called from within BEAM itself (via the L3 Binning Processor Tool) or standalone via the `binning.sh` command. The standalone command will by default open the GUI, but it can also be called via an XML file. The XML file can be created from the GUI and has the following format:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<RequestList>
    <Request type="BINNING">
        <Parameter name="process_type" value="init" />
        <Parameter name="database" value="/home/cluster/lbrinckm/l3_database-1.b
indb" />
        <Parameter name="lat_min" value="-90.0" />
        <Parameter name="lat_max" value="90.0" />
        <Parameter name="lon_min" value="-180.0" />
        <Parameter name="lon_max" value="180.0" />
        <Parameter name="log_prefix" value="l3" />
        <Parameter name="log_to_output" value="false" />
        <Parameter name="resampling_type" value="binning" />
        <Parameter name="grid_cell_size" value="55.0" />
        <Parameter name="band_name.0" value="CloudTopHeight" />
        <Parameter name="bitmask.0" value="CloudTopHeight &gt;= 1000 and sun_ele
v_nadir &gt;= 10.0" />
        <Parameter name="binning_algorithm.0" value="Arithmetic Mean" />
        <Parameter name="weight_coefficient.0" value="1.0" />
    </Request>
    <Request type="BINNING">
        <Parameter name="process_type" value="update" />
        <Parameter name="database" value="/home/cluster/lbrinckm/l3_database-1.b
indb" />
        <Parameter name="log_prefix" value="l3" />
        <Parameter name="log_to_output" value="false" />
        <InputProduct file="/home/cluster/lbrinckm/Data/stereo/beamtest/200304/1
200/05668.dim" />200/05824.dim" />

...
        <InputProduct file="/home/cluster/lbrinckm/Data/stereo/beamtest/200304/1
```

```
200/05825.dim" />
    </Request>
    <Request type="BINNING">
        <Parameter name="process_type" value="finalize" />
        <Parameter name="database" value="/home/cluster/lbrinckm/l3_database-1.b
indb" />
        <Parameter name="delete_db" value="true" />
        <Parameter name="log_prefix" value="l3" />
        <Parameter name="log_to_output" value="false" />
        <Parameter name="tailoring" value="true" />
        <OutputProduct file="/home/cluster/lbrinckm/l3_out.dim" format="BEAM-DIM
AP" />
    </Request>
</RequestList>
```

The output file is in this example set to BEAM-DIMAP, but it is also possible to specify GeoTiff.[2]

4 Implementation and Compilation

This section covers details of the module's implementation. Its understanding is not required to run the module.

4.1 Package Organisation

Java supports software modularisation through its package concept. All packages in the MSSL Stereo Matcher module are in sub-packages of `uk.ac.ucl.mssl.climatephysics`. Below this the following packages are organised:

`*.beam.*`: containing code depending and extending the functionality of BEAM. These generally contain pairs of files, one containing the core functionality, the other, named *Action, code required for GUI integration.

- **`*.beam.atsr`:** routines for ATSR data.
- **`*.beam.imaging`:** generalised BEAM imaging routines.
- **`*.beam.stereomatcher`:** generalised stereo matching routines for BEAM.

`*.imaging`: BEAM independent imaging routines, such as extensions to Java Advanced Imaging and Kernel Implementations.

`*.stereomatcher`: BEAM independent stereo matching routines, such as a camera models and disparity set generators

`*.utilities`: generic utilities for interpolation and array processing.

Where time allowed a sub-package `tests` contains unit tests based on the JUnit testing framework.

4.2 Stereo Matcher Inheritance

The M4 and M5 stereo matchers only vary in their calculation of the disparity sets: while M5 tests for every pair of possible disparities with the search window, the M4 algorithm only tests those disparities that are most dominant. The different strategies to implement the disparity sets are implemented in

[2]However, the georegistration is not entirely accurate for GeoTiffs.

package `uk.ac.ucl.msl.climatephysics.stereomatcher` as classes derived from `DisparitySetGenerator`. M5 utilises the `DenseDisparitySetGenerator`, while M4 utilises `CorrelationShiftDisparitySetGenerator`. M4 is technically derived from M5, overriding the `generateDisparitySet` method.[3]

4.3 Image Tiling

BEAM is built upon the functionality of the Java Advanced Imaging library. One of its facilities is an automatic tiling mechanism that allows to process very large images effectively as the entire image is not required to be in memory at the same time. Scene-based versions of the M4 stereo-matcher showed significant edge-effects resulting from the applications of kernels, but as orbit-based data is now available these edge-effects can be almost entirely eliminated in the along-track direction, leaving only edge-effects along the outer edges of the orbit. For this the tiling mechanism in BEAM is influenced so that, depending on parameter settings for kernels, individual scenes are overlapped as much as is required to build a continuous product. This is achieved by setting the preferred tile size to less then 512 * 512 pixels using the `setPreferredTileSize` method during operator initialisation.

Any filter will result in increased speed only for those tiles that are entirely filtered out. These cases are detected before the stereo-matcher begins work. However, for partly masked scenes the disparities will first be calculated and then masked out as it is not possible to compute the image correlation based on a sparsely filled image.

4.4 Compiling the Module

As Java is a statically typed compiled language, future releases of BEAM are likely to require a recompilation of the library. Rebuilding the module requires the same setup as building BEAM from source, documentation on this can be found at `http://www.brockmann-consult.de/beam-wiki/display/BEAM/Build+from+Source`. The source code of the library is supplied as compressed UNIX tar archive (as we never manged to set up a version control system at MSSL), which can be downloaded from `ftp://ftp.mssl.ucl.ac.uk/pub/imaging/lmb2/mssl-stereomatcher-0.8-src.tgz`. The archive file must then be extracted.

```
tar -xvf mssl-stereomatcher-0.8-src.tgz
```

To compile, a Java development environment is required as detailed in the BEAM documentation. The source archive is supplied together with a `build.xml` build instructions that can be read by the Ant tool (`http://ant.apache.org/`. The build instructions in the `build.xml` file will require some simple changes to reflect the location of your BEAM installation. After this the Java code can be simply compiled:

```
ant build.xml
```

This should produce output similar to this:

```
Buildfile: build.xml

build-subprojects:

init:
    [mkdir] Created dir: c:\Users\Ludwig\Documents\dev\mssl-stereomatcher\bin
     [copy] Copying 2 files to c:\Users\Ludwig\Documents\dev\mssl-stereomatcher\
bin

build-project:
     [echo] mssl-stereomatcher: c:\Users\Ludwig\Documents\dev\mssl-stereomatcher
```

[3]A better implementation would have used policy or strategy objects.

```
\build.xml
    [javac] Compiling 35 source files to c:\Users\Ludwig\Documents\dev\mssl-ster
eomatcher\bin
    [javac] Note: c:\Users\Ludwig\Documents\dev\mssl-stereomatcher\src\uk\ac\ucl
\mssl\climatephysics\imaging\SqrtDescriptor.java uses or overrides a deprecated
API.
    [javac] Note: Recompile with -Xlint:deprecation for details.

build:

BUILD SUCCESSFUL
Total time: 3 seconds
```

4.5 Cluster Processing and Climatologies

The steps described in the previous sections have detailed how to process an individual orbit on a single machine. In this section the steps to produce climatologies from a larger set of ATSR data are detailed.

4.5.1 Cluster Processing

BEAM does not offer dedicated cluster processing, instead it was chosen to wrap the processing of a single day of ATSR data into one script. This script is somewhat tied to the environment at RAL, yet an adaptation to other environments would not be difficult.

To facilitate the execution of the script for different days on a varying number of machines Parallel Python was used. Parallel Python provides simple means of transfering any Python script for execution to a remote machine – and as Python can call shell scripts, it is not limited to Python code alone. Parallel Python implements a master/slave pattern of remote execution and the library provides means of dynamically discovering where slave workers are available. The slave worker is simply a process waiting to accept commands on a remote machine: it will accept one command (actually one command per processor on its machine) at a time, executes it and then awaits further commands. The master process simply discovers which slaves are around and dispatches commands in order.

The slave worker process is implemented by the `ppserver.py` script provided by Parallel Python. Running `ppserver.py -a` will start such a process which will listen on a port for incoming commands.

Once the slave workers are present, commands or jobs can be dispatched to them. The script `msslparallel` dispatches a month of days to the waiting workers.[4]

A utility script to run the same command within the RAL cluster is `msslclusterrun`. `msslclusterrun` takes two parameters: a Windows .ini style configuration file plus the name of a section of the ini file, e.g.

```
msslclusterrun -s startppservers config
```

`msslclusterrun` will then once and only once prompt for a password for the remote machines, which is not anywhere on disk and will not show up in any logs.[5].

References

Marie-Ange Denis, Jan-Peter Muller, and Hermann Mannstein. ATSR-2 camera models for the automated stereo photogrammetric retrieval of cloud-top heights: Initial

[4]Adaptation to a year of data would be simple.

[5]If the password supplied is wrong `msslclusterrun` will not prompt again and simply ungracefully exit. There remains a bug that sometimes the command appears unable to log in, this has not been fixed. Restarting the command is usually fine.

assessments. *International Journal of Remote Sensing*, 9:1939–1955, 2007. *doi* : 10.1080/01431160600641723.

Catherine Moroney, Ákos Horváth, and Roger Davies. Use of stereo-matching to coregister multiangle data from MISR. *IEEE Transactions On Geoscience And Remote Sensing*, 40 (7):1541–1546, July 2002. *doi* : 10.1109/*TGRS*.2002.801146.

J.-P. Muller, M.-A. Denis, R. D. Dundas, K. L. Mitchell, C. M. Naud, and H. Mannstein. Stereo cloud-top heights and cloud fraction retrieval from ATSR2. *International Journal of Remote Sensing*, 9:1921–1938, 2007. *doi* : 10.1080/01431160601030975.