Envisat MERIS

Geometry Handbook

	name *function* *company*	*date* *signature*
prepared by	Serge RIAZANOFF *Engineer* VisioTerra *Serge.Riazanoff@visioterra.com*	
checked by	Stéphane MBAYE *Engineer* GAEL Consultant *Stephane.Mbaye@gael.fr*	
authorized by	Christophe DEMANGE *Director* GAEL Consultant *Christophe.Demange@gael.fr*	
approved by	Steven DELWART *Project Manager* ESA-ESTEC *Steven.Delwart@esa.int*	

DOCUMENT STATUS SHEET

Issue	Date	Comments	Author
1.0	16/08/2004	Draft 1 - Creation of the document – Orthorectification results	S. Riazanoff
1.0	19/08/2004	Draft 2 – Level 1B model, performances, disparity analysis	S. Riazanoff
1.0	19/08/2004	Draft 3 – Orthorectification application	S. Riazanoff
1.0	23/08/2004	Comparison BEAM vs. MERSYN	S. Riazanoff
1.1	29/10/2004	Comments Marc BOUVET on 20/09/2004	S. Riazanoff
1.2	08/11/2004	Release 02.04 of MERSYN – Appendix C	S. Riazanoff
		Benchmark – Appendix D	S. Riazanoff
1.3	16/11/2004	Section 5 – GETASSE30 Quality Control	S. Riazanoff
1.4	03/01/2005	Comments of Norman Fomferra – Eq. 2, 6a and 6b, Appendix E, MERSYN program version 02.05	S. Riazanoff
1.5	06/05/2005	Completion of document – Chapter 5.2. Comparison of GETASSE30 with GPS points	S. Riazanoff

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Purpose of this document

This document describes an accurate method to orthorectify MERIS 1B scenes or segments. This project has been performed on behalf the European Space Agency under the ESTEC contract 17944/03/NL/CO with GAEL Consultant, which has subcontracted the works to VisioTerra.

The overall contract foresees two work packages:

- WP1 - Envisat MERIS Geometry handbook describing the orthorectification algorithm providing examples of code.
- WP2 – SRTM / DEM Digital Elevation Model Quality Assessment.

Algorithms laid down in this document apply to both Full Resolution (FR) and Reduced Resolution (RR) MERIS level 1B products.

fig. 1 - Artist view of Envisat satellite.

1.2 Document overview

- Chapter 1 is the present section applicable to the overall document.
- Chapter 2 describes the MERIS level 1B geometry.
- Chapter 3 describes the orthorectification process.
- Chapter 4 gives results of the orthorectification process.
- Chapter 5 tests the quality of the GETASSE30 DEM.
- Appendix A includes the header information of the MERIS scene being tested.
- Appendix B includes the logs of the orthorectification execution.
- Appendix C provides the C-language code of MERSYN application.
- Appendix C provides the C-language code of MERSYN application.

- Appendix D details the benchmark used to orthorectify a scene over Italy.
- Appendix E logs the orthorectification correction performed on first and last pixels of some lines.
- Appendix F details the differences between GTASSE30 30 and geodesic point elevations.

1.3 Applicable documents

A-1	PO-SW-ESA-GS-1393	*ESTEC Contract 17944/03/NL/CO* *Statement Of Work* Issue 1, 19/11/2003 ESA - ESTEC
A-2	e-mail Steven DELWART	*Use of GETASSE30 DEM* 22/10/2004 Steven DELWART - ESA - ESTEC ..\management\mail_20041022_Delwart_GETASSE30 data available.txt

1.4 Reference documents

This section describes the related documents and applied conventions to be considered within the present document.

R-1	PO-RS-MDA-GS-2009 (1)	*Envisat-1 Products Specifications* *Volume 1 - Introduction* Issue 3, Revision L – 30/01/2001 ESA – ALCATEL SPACE
R-2	PO-RS-MDA-GS-2009 (11)	*Envisat-1 Products Specifications* *Volume 11 – MERIS Products Specification* Issue 3, Revision J – 12/05/2003 ESA – ALCATEL SPACE http://earth.esa.int/pub/ESA_DOC/ENVISAT/MERIS/Vol11_Meris_3j.pdf
R-3	PO-TN-MEL-GS-0002	*MERIS Level 1 Detailed Processing Model,* *Parameters Data List* Issue 6, Revision 1 – 28/03/2003 ACRI ..\..\A001_REFERENCE_DOCUMENTS\ENVISAT\PO-TN-MEL-GS-0002_DPM1b_i6r1.pdf
R-4	PO-TN-MEL-GS-003	*MERIS Input / Output Data Definitions* Issue 6 revision 1 – 23/03/2003 ACRI
R-5	SRTMvsACE_draft_v3.doc	*Intercomparison of ACE and SRTM30 global digital elevation model at global and regional scale* Issue – Revision 3 – 08/10/2004 Marc BOUVET – ESA/ESTEC \\Leo\leo_disks\leo_2\P194_ESTEC_MERIS_GEOMETRY_HANDBOOK\DEM_GETASSE30\doc\SRTMvsACE_draft_v3.doc

R-6 Readme_with_images.doc — *GETASSE30 - Global Earth Topography and Sea Surface Elevation at 30 arc second resolution*
Issue -, Revision -, 20/10/2004
Marc BOUVET – ESA/ESTEC
\\Leo\leo_disks\leo_2\P194_ESTEC_MERIS_GEOMETRY_HANDBOOK\DEM_GETASSE30\doc\Readme_with_images.doc

R-7 e-mail Norman FOMFERRA — *Comments on equations 6a and 6b*
17/12/2004
Norman FOMFERRA – Brockman Consult
..\management\mail_20041217_Fomferra_MERIS_Geometry_Handbook.txt

Most of the fields of the MERIS 1B products required in for the orthorectification are given in the product specification document R-2 but may also be found at the following Web addresses:

- MERIS FR 1B - http://envisat.esa.int/dataproducts/meris/CNTR6-2-1.htm#eph.meris.merisdf.1p.MER_FR__1P,
- MERIS RR 1B - http://envisat.esa.int/dataproducts/meris/CNTR6-2-2.htm#eph.meris.merisdf.1p.MER_RR__1P.

1.5 Abbreviations and Acronyms

This section controls the definition of all abbreviations and acronyms used within this document. Special attention has been paid to adopt abbreviations, acronyms and their definitions from international standards as ISO, ANSI or ECSS.

Abbreviations and acronyms specific to Envisat/MERIS may be found at address http://envisat.esa.int/dataproducts/meris/CNTR5-1.htm.

ACE	Altimetry Corrected Elevations
ADSR	Annotation Data Set Record
ANSI	American National Standards Institute
DEM	Digital Elevation Model
ECSS	European Cooperation for Space Standardization
EGM96	Earth Gravity Model 1996
EO	Earth Observation
ETRS89	European Terrestrial Reference System 1989
FR	Full Resolution
GETASSE30	Global Earth Topography and Sea Surface Elevation at 30 arc second resolution
ISO	International Organization for Standardization
ITRS	International Terrestrial Reference System
ITRF	International Terrestrial Reference Frame
MDS	Measurement Data Set
MDSR	Measurement Data Set Record
MSS	Mean Sea Surface
NDVI	Normalised Difference Vegetation Index

RR	Reduced Resolution
SRTM	Shuttle Radar Topography Mission
TBC	To Be Confirmed
TM	Thematic Mapper (instrument of Landsat)
WGS84	World Geodetic System 1984

1.6 Definitions

This section provides the definition of all common terms used within this document. Special attention has been paid to adopt definitions from international standards as ISO, ANSI or ECSS.

ESA also provides the following glossaries (only those interesting for this document have been selected):

- Geometry http://envisat.esa.int/dataproducts/meris/CNTR5-2-1.htm,
- Optics http://envisat.esa.int/dataproducts/meris/CNTR5-2-9.htm,
- Product http://envisat.esa.int/dataproducts/meris/CNTR5-2-10.htm,

geocoded	An image (or more generally any EO data) is geocoded if a simple relation exists giving the geodetic coordinates (λ,φ) or (X,Y) of the attached geodetic system, from the coordinates of any point (i,j) of the image.

2 MERIS LEVEL 1B GEOMETRY

Scope of this section is to quickly describe the geometry of the Envisat MERIS level 1B products. A more detailed description is to be searched in document R-3. Content of this document is also presented at http://www.envisat.esa.int/dataproducts/meris/CNTR2-6-1.htm address.

2.1 MERIS Instrument

MERIS instrument is flown onboard the Envisat satellite (see fig. 2).

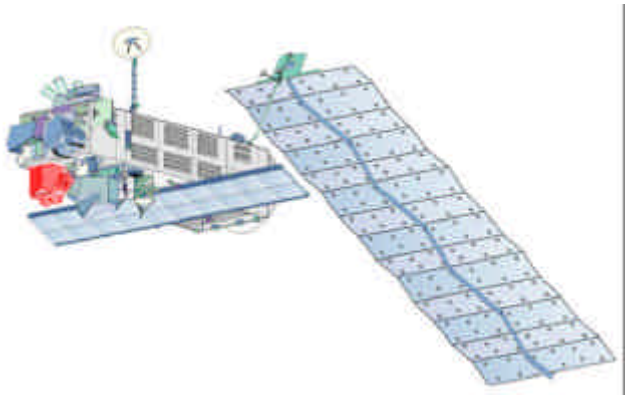


fig. 2 - MERIS instrument (red) on Envisat
(http://envisat.esa.int/dataproducts/meris/CNTR3-1-1.htm)

This push-broom instrument acquires top of atmosphere radiances in 15 visible and NIR bands and is an assembly of five (5) cameras, sometimes called *modules* because each camera is included within a module sub-system (see fig. 3).

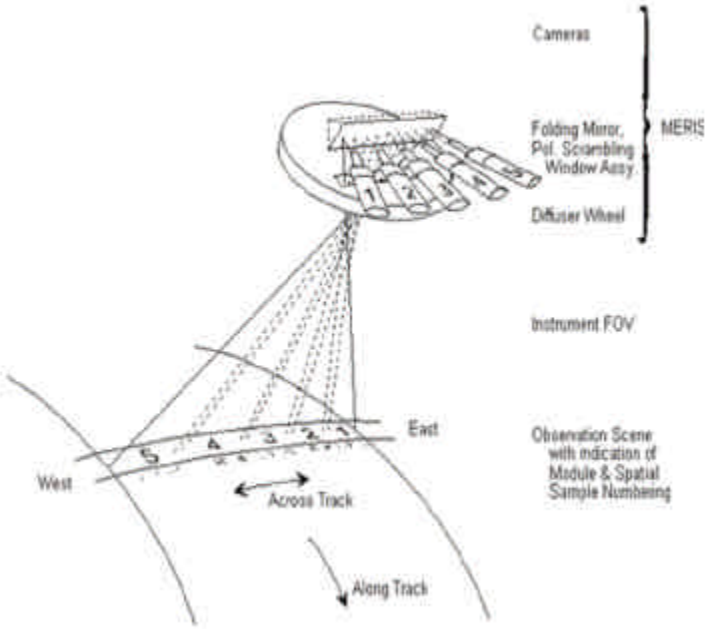


fig. 3 - MERIS FOV, camera tracks, pixel enumeration and swath dimension
http://envisat.esa.int/dataproducts/meris/CNTR1-1-3.htm

2.2 Processing levels

Processing from Level 0 to Level 1B includes the following tasks:

- source data packet extraction,
- saturated pixels replacement and flagging,
- radiometric processing,
- stray light correction,
- **geo-location,**
- pixel classification,
- external data assimilation,
- formatting.

Figure fig. 4 here below illustrates the workflow.

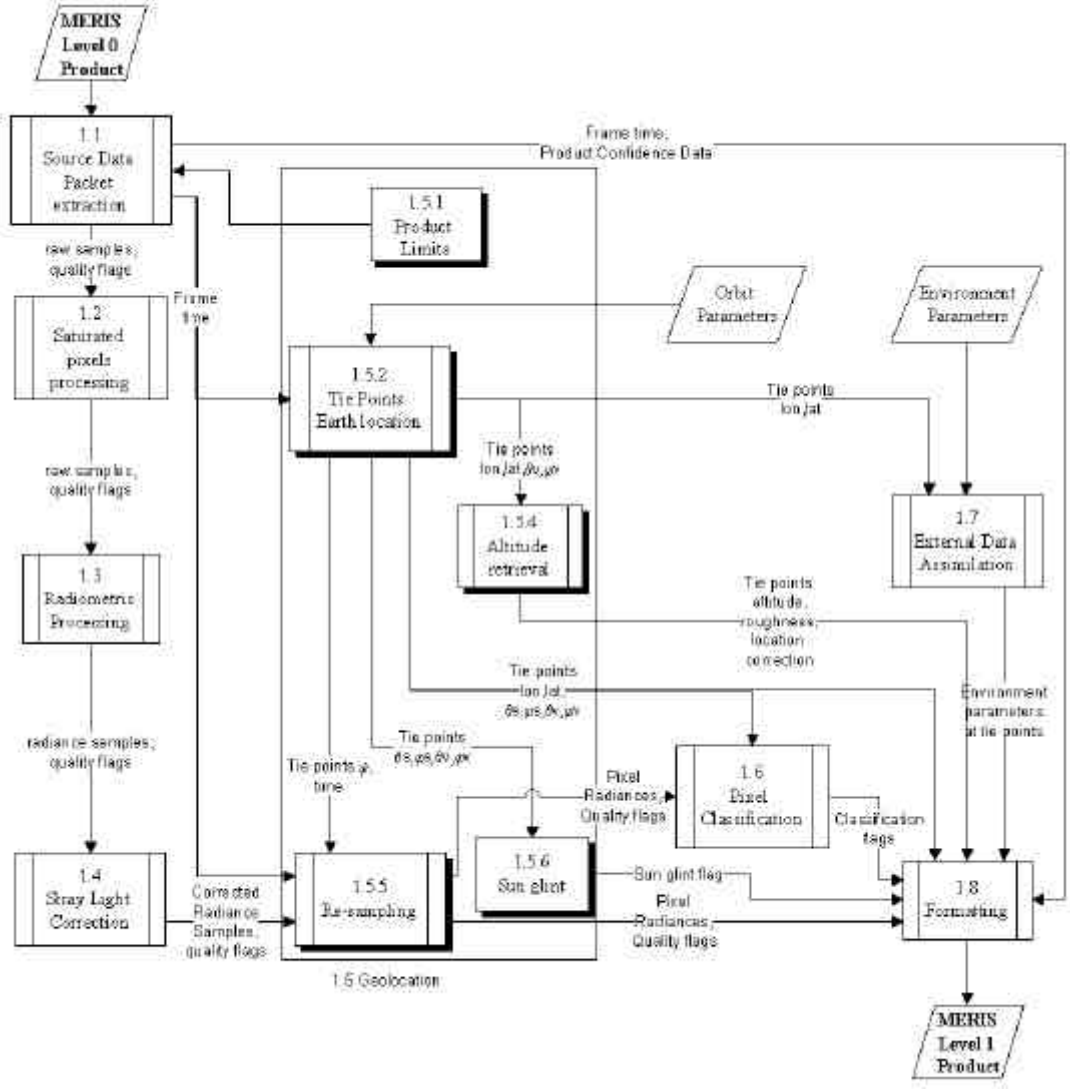


fig. 4 - Functional Breakdown for Level 1B processing algorithm (R-3).

MERIS Level 1B product grid and tie-point grid

The level 1B product grid is a geolocated system defined by the vertical projection of the satellite trajectory over the Earth. (fig. 5) defined by the velocity vector and by its perpendicular plane crossing the Earth.

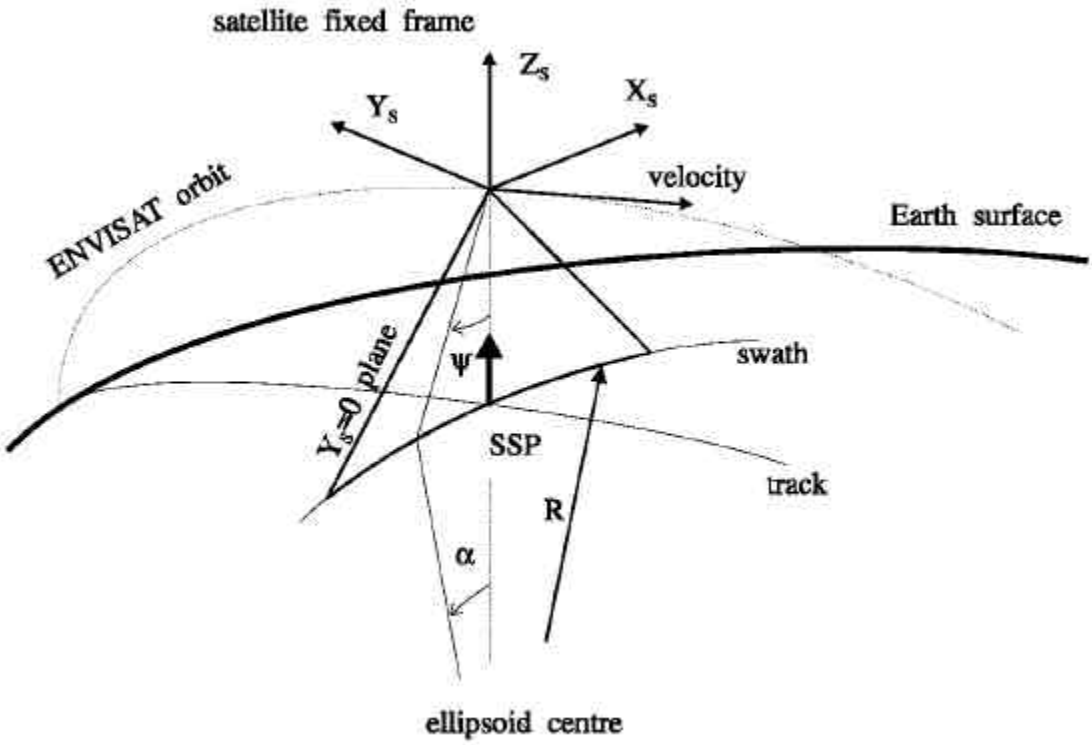


fig. 5 - Satellite fixed coordinate system (R-3).

The document "MERIS Level 1 Detailed Processing Mode" (R-3) says:

> *"In order to simplify product handling, the MERIS radiance samples are re-located by nearest neighbour interpolation to the MERIS product grid, which has the following characteristics (FR grid) :*
>
> - *central column : sub-satellite point track on Earth;*
> - *line orientation : perpendicular to spacecraft velocity, projected on Earth;*
> - *columns spacing : fixed for one product, 260 m (with very small variations);*
> - *number of columns : 4481;*
> - *line spacing : variable with time and orbit altitude, fixed by the MERIS frame time of 0.044s (mean » 292 m).*
>
> *The RR-grid is a 4x4 sub-sampled version of that grid." (R-3)*

To provide geodetic information (latitude, longitude, DEM elevation and roughness), but also information relative to the acquisition conditions (sun incidence angles, viewing angles, wind directions, pressure, ozone, humidity) a **tie-point grid** is superimposed over the MERIS level 1B product grid (see fig. 6). These tie-point grid values are stored in the MERIS 1B product within the Annotation Data Set (ADS) / Tie Points Location & Corresponding Auxiliary Data (LADS).

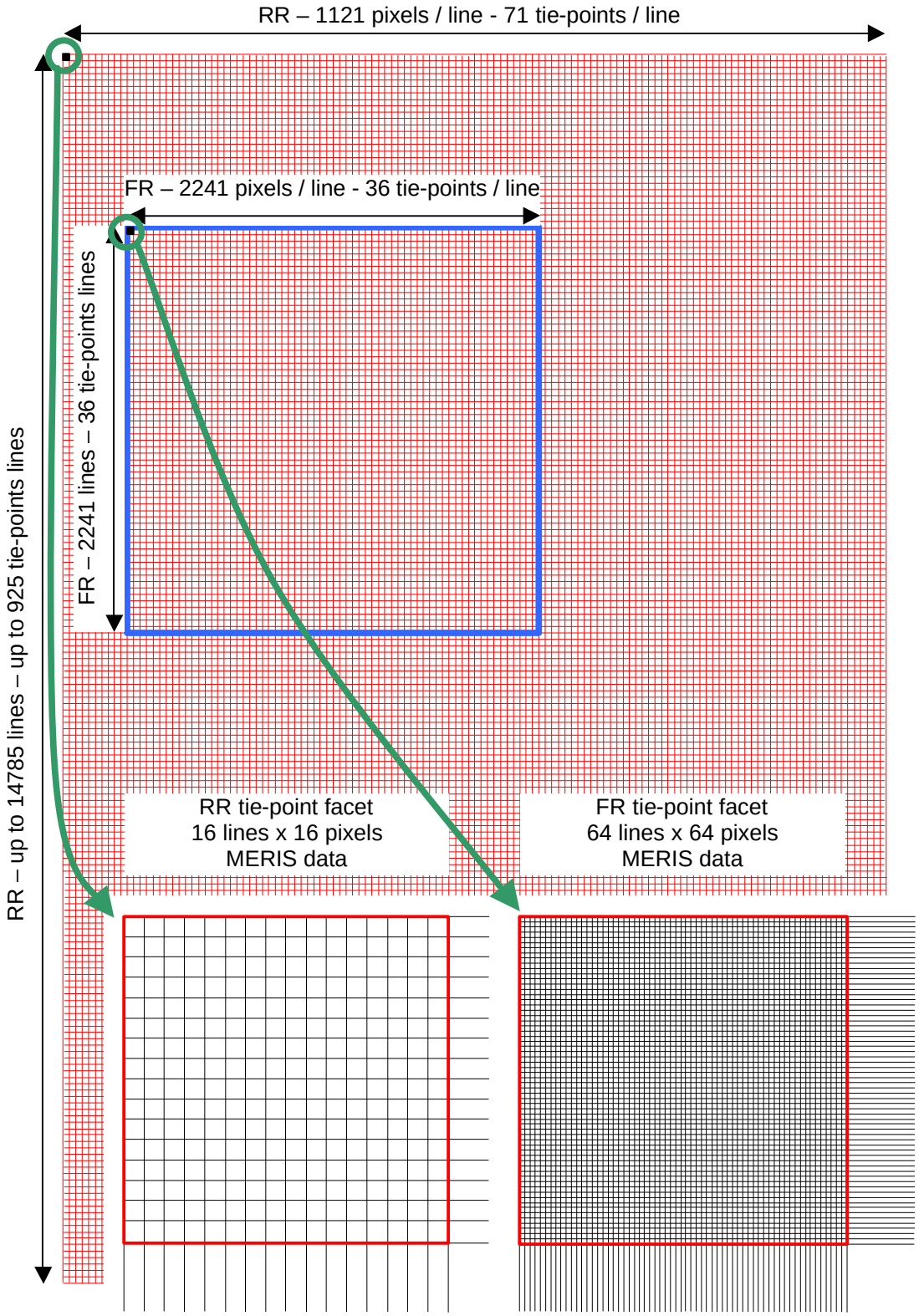


fig. 6 - MERIS level 1B grid and tie-point grid.

The tie-point grid "contains":

- 64 lines x 64 pixels MERIS data for the FR product, and
- 16 lines x 16 pixels MERIS data for the RR product.

The size of products and the number of tie-points to be found in the Annotation Data Set are given in the table 1 below.

Product	MERIS grid		Tie-point grid	
	Line number	pixel number / line	annotation data set (ADSR) number	samples number / ADSR
MERIS RR 1B	≤ 14785(1)	1121	≤ 925	71
MERIS FR 1B	2241	2241	36	36
MERIS FS 1B(2)	4481	4481	71	71

(1) A full segment may match up to 43.5 minutes (a sunlight orbit) representing up to 17400 km on ground.

(2) The full resolution Full Swath will be available for the next release of the processor and all the values provided here are to be confirmed.

table 1 - Size and number of tie-points in products.

Along-track tie-points

The constant along-track sampling of 0.044 seconds produces periodic MERIS frames that contain measurements resampled on the MERIS grid. Each DF MERIS frames (DF=64 for FR and DF=16 for RR), a "tie frame" is computed giving the position of the grid along the MERIS swath (see fig. 7).

DF (ΔF in fig. 7): tie-frame spacing.

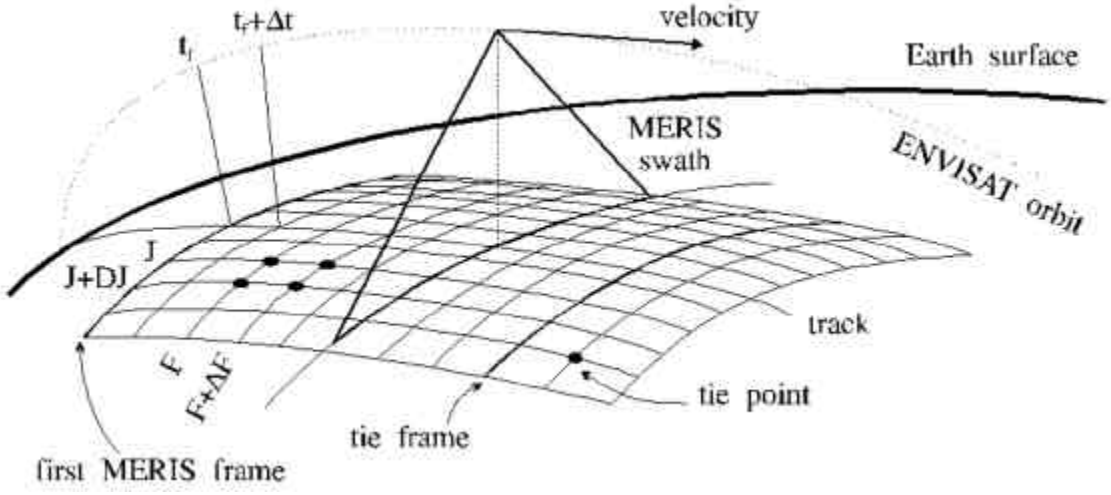


fig. 7 - MERIS frames and tie frames (R-3).

Across-track tie-points

Tie-points along the MERIS swath are sampled in order to be evenly spaced (in distance) along the swath (see fig. 8).

DJ: tie-points column spacing.

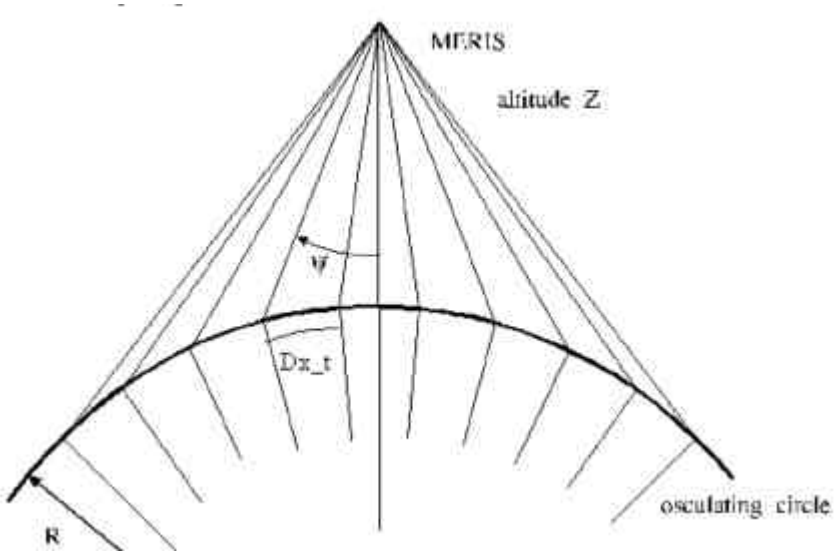


fig. 8 – tie-points pointing direction (R-3).

Bi-linear interpolation within « tie-point facets »

Let (F,J), (F+DF,J), (F,J+DJ) and (F+DF,J+DJ) the coordinates in the MERIS Level 1B product grid of four adjacent tie-points (see fig. 9), any ancillary value X(F,J) given at the tie-point location may be interpolated within the facet at the location (f,j) using the following formula:

$$X(F+f,J+j) = \left(\frac{DJ-j}{DJ}\right)\times\left[\left(\frac{DF-f}{DF}\right)\times X(F,J) + \left(\frac{f}{DF}\right)\times X(F+DF,J)\right] + \left(\frac{j}{DJ}\right)\times\left[\left(\frac{DF-f}{DF}\right)\times X(F,J+DJ) + \left(\frac{f}{DF}\right)\times X(F+DF,J+DJ)\right] \quad \text{Eq. 1}$$

Where:

X() is the measurement stored on each tie-point (latitude, longitude, DEM altitude, DRM roughness, DEM latitude correction, DRM roughness, sun zenith, sun azimuth, viewing zenith, viewing azimuth, zonal winds, meridional winds, mean sea level pressure, total ozone, relative humidity).

(F,J) are the coordinates of the frame (along-track) and column (across-track) expressed in the product grid reference system (i.e. the line index and column index of the level 1B image).

DF is the number of product lines (frames) between two consecutive tie-points along-track.

DJ is the number of product pixels between two consecutive tie-points across-track.

(f,j) are the coordinates within the tie-point facet ($f \in [0,DF[$ and $j \in [0,DJ[$). These coordinates may be float and not necessarily integer.

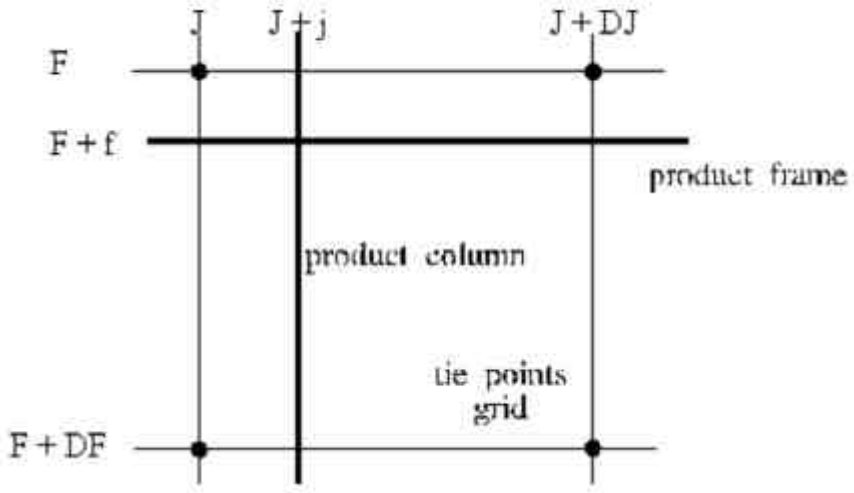


fig. 9 - Interpolation within a "tie-point facet" (R-3).

Parallax estimate at tie-points

As shown in fig. 10, the parallax may be estimated knowing the altitude and viewing angle at any point of the product grid. In particular, the geodetic correction terms (dlat,dlon) are said being computed by the formula below and stored in the Tie Points ADSR (see R-3, section 8.3.3, page 8-28):

$$dx = Z_{F,J} \times \tan(\theta_{vF,J})$$
$$dlat_{F,J} = \frac{dx \times \cos(\varphi_{vF,J})}{R_e}$$
$$dlon_{F,J} = \frac{-dx \times \sin(\varphi_{vF,J})}{R_e \times \cos(\phi_{F,J})}$$

Eq. 2

Where:

dx	is the modulus of the parallax error.
$dlat_{F,J}$	is the latitude correction of tie-point (F,J).
$dlon_{F,J}$	is the longitude correction of tie-point (F,J).
$Z_{F,J}$	is the DEM elevation of tie-point (F,J).
$\theta_{vF,J}$	is the zenith of the viewing angle of tie-point (F,J).
$\varphi_{vF,J}$	is the azimuth of the viewing angle of tie-point (F,J).
$\phi_{F,J}$	is the geodetic latitude of tie-point (F,J).
R_e	is the mean Earth radius (R_e = 6 370 997.0 metres)

Examples of tie-point values and explanation relative to the way data are computed are provided in APPENDIX E. Observation of correction values for latitude and longitude shows that equ. 2 is not strictly exact (see eq. 2' in APPENDIX E).

Slightly different formulae (see eq. 6a and 6b) are proposed in sub-section "Compute and apply the geodetic correction" of section 3.3.

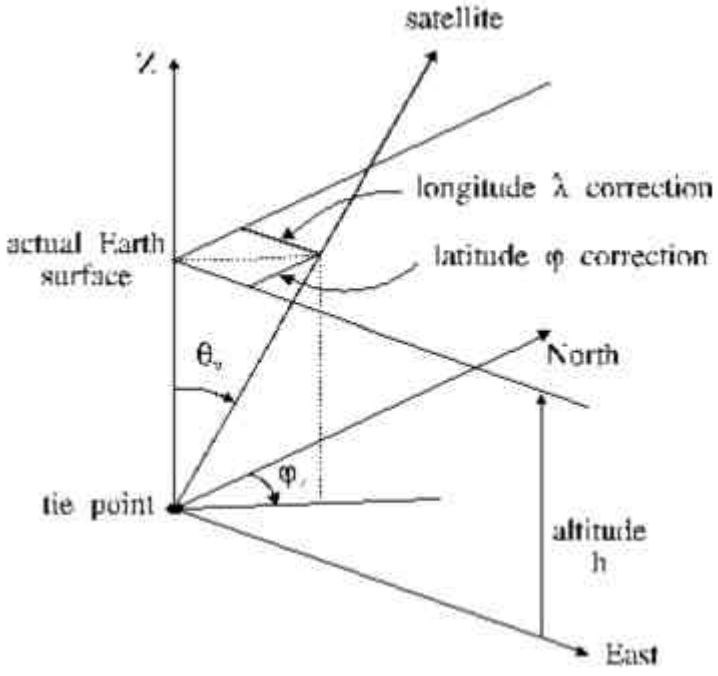


fig. 10 - Parallax estimation principle (R-3).

Projection of MERIS pixels to the level 1B product grid

After radiometric correction, each MERIS pixel P_k of the frame f is copied into the MERIS Level 1B product grid at the location of coordinates (f',j) that are the nearest integer of the projection coordinates within the tie-point facet [F,F+DF]x[J,J+DJ].

These projection coordinates are computed from the (θ_k,ψ_k) viewing vector of point P_k.

The viewing vector it-self is computed from propagated ephemeris and attitude values and from the nominal values of the CCD viewing vectors.

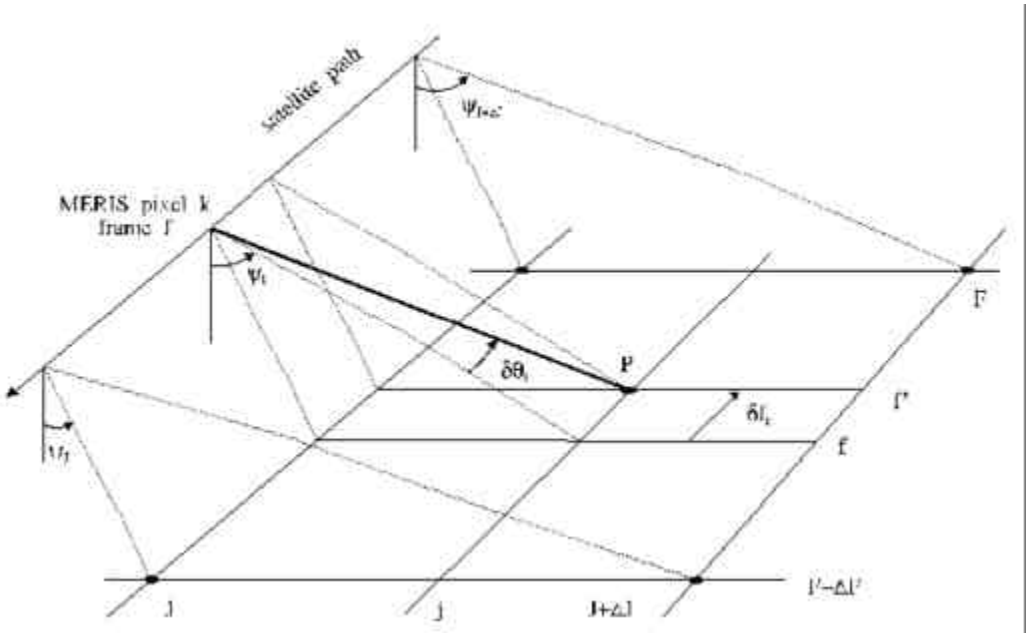


fig. 11 - MERIS pixel location (R-3).

2.3 MERIS FR and RR level 1B products

The overall characteristics of MERIS level 1B products are shown in the table here below, enabling the comparison between Full Resolution (FR) and Reduced Resolution (RR) products.

PRODUCT ID	**MER_FR__1P**	**MER_RR_1P**
NAME	Level 1b Full Resolution	Level 1b Reduced Resolution
DESCRIPTION	MERIS product calibrated and geolocated. Produced on request only Floating scene concept for distribution	MERIS product calibrated (radiance) and geolocated. Produced Systematically
COVERAGE	575 km x 575km	1150 km x 17500 km
THROUGHPUT	6 PDHS-E 3 PDHS-K	1 per orbit
GEOMETRICAL SAMPLING	300 x 300 meters resampled in a "pseudo satellite" projection along track	1.2 km x 1.2 km resampled in a "pseudo satellite " projection along track
SEGMENT SIZE	157 bytes per scene	532 Mbytes
RADIOMETRIC RESOLUTION	1.14 x 10^{-1} w/m^2/sr/µm@442.5nm SSP with Q_S=60°	2.85 x 10^{-2} w/m^2/sr/µm@442.5nm SSP with Q_S =60°
RADIOMETRIC ACCURACY	from 400 to 900 nm < 2% from 900 to 1050 nm < 5%	from 400 to 900 nm < 2% from 900 to 1050 nm < 5%
DATA SET	MPH SPH ADS MDS	MPH SPH ADS MDS
ANNOTATION DATA	Orbit state vector, Time correlation parameters, Latitude, Longitude, altitude and topographic corrections Sun azimuth, Sun elevation, view azimuth, view elevation Resampled ECMWF data: Mean Sea Level pressure, Total column ozone, Total column water vapour, Wind speed,	Orbit state vector, Time correlation parameters, Latitude, Longitude, altitude and topographic corrections Sun azimuth, Sun elevation, view azimuth, view elevation Resampled ECMWF data: Mean Sea Level pressure, Total column ozone, Total column water vapour, Wind speed,
NOTES	On demand dissemination of multiple of scenes (575km x 575km) or (296km x 296km)	Produced systematically. On demand dissemination of multiple of scenes (1150 km x 1150 km)

table 2 - MERIS FR and RR Level 1B product descriptions
(http://envisat.esa.int/instruments/meris/data-app/prodspread.html)

3 ORTHORECTIFICATION APPLICATION

Orthorectified images are corrected of the acquisition conditions (viewing geometry, platform attitude, Earth rotation and of the relief effects (parallax). Such a corrected images are perfectly superimposable whatever being their acquisition dates.

One of the principal applications of such a superimposability is the possibility to process mosaicking and more simply synthesis. For example, a synthesis of all the MERIS RR 1B segments acquired on March and April 2003 has been created by GAEL Consultant in June 2003.

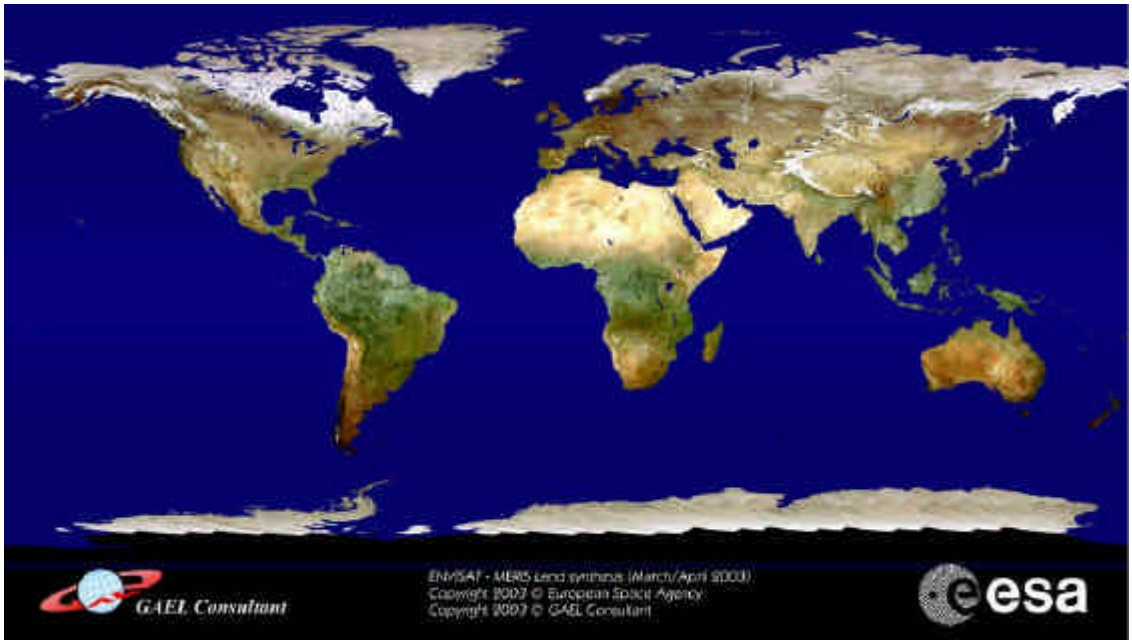


fig. 12 - MERIS synthesis March-April 2003

This project has been the opportunity to develop a first version of a synthesis process, called MERSYN. The present contract has given the possibility to refine MERSYN application extending the orthorectification process to MERIS FR.

This is the reason reader will sometimes encounter references to the synthesis process, up to the name of the provided code (MERSYN for MERIS synthesis). But the synthesis it-self is marginal; most of the code is dedicated to orthorectification that is the challenging activity.

3.1 Overall architecture

MERSYN is able to orthorectify a MERIS Level 1B scene or segment and to add it over an existing synthesis. Merging of a new MERIS pixel over an existing synthesis may be performed according to two different “selection algorithms”: “Minimum radiance” or “NDVI maximum”. The number of parameters for these two behaviour is not the same providing two use cases.

When no synthesis is provided in input, MERSYN only orthorectifies the segment or scene. This is the third use case.

Use case 1 – Minimum radiance synthesis

In this use case the selection algorithm (« -sal » parameter) has the value "Minimum radiance" meaning that for each band b the MERIS pixel R(b,i,j) will replace the synthesis pixel $S_{n-1}(b,i,j)$ only if MERIS pixel is inferior.

$S_n(b,i,j) \leftarrow$

$\text{Min}[R(b,i,j), S_{n-1}(b,i,j)]$

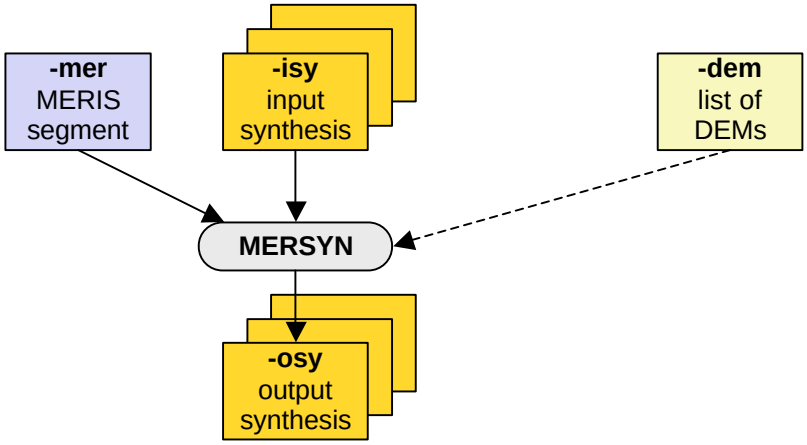


fig. 13 - Minimum radiance synthesis parameters.

Use case 2 – Maximum NDVI synthesis

In this use case the selection algorithm (« -sal » parameter) has the value "Maximum NDVI" (Normalised Difference Vegetation Index).

In this algorithm, all the bands b of a MERIS pixel R(b,i,j) will replace the previous synthesis value $S_{n-1}(b,i,j)$ only if the new NDVI is greater than the previous one $NDVI_{n-1}(i,j)$.

$$NDVI_n(i,j)=\frac{R(13,i,j)-R(7,i,j)}{R(13,i,j)+R(7,i,j)}$$

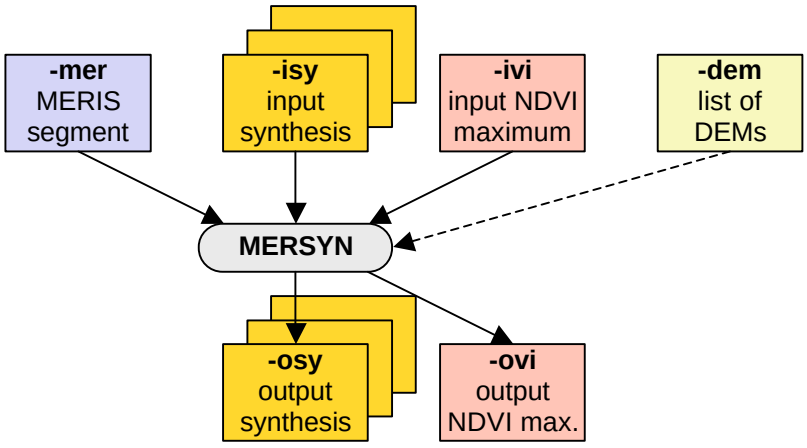


fig. 14 - Maximum NDVI synthesis parameters.

Use case 3 – Orthorectified mapping

In this use case, only the mapping is performed. The output image may be orthorectified using MERIS tie-points elevation values or one (or more) DEM(s) provided as input parameters.

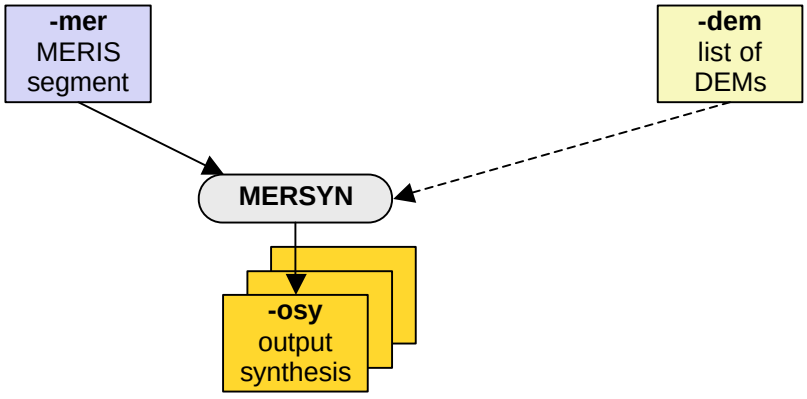


fig. 15 - Orthorectified mapping parameters.

Digital elevation data

MERSYN may:

- no process any orthorectification ("-ort" parameter set to "NO"),
- uses elevation data found in tie-points auxiliary data of the product (no values given to "-dem" parameter),
- uses elevation found within the provided DEM(s) ("-dem" parameter).

More than one DEM may be provided. In such a case, for each pixel of the output geocoded image MERSYN will search for the corresponding elevation data within the first DEM in the list. If the point is outside the DEM, or if a background value (meaning no elevation) is found within this DEM, MERSYN will attempt to get elevation data from the second DEM in the list, and so on…

When more than one DEM are provided, it is recommended that all the DEM belongs to the same series; for example tiles of ACE, SRT30 or GTOPO30. When such is not the case, it is recommended to set the DEM having the greater accuracy as first DEM in the "-dem" list.

Workflow

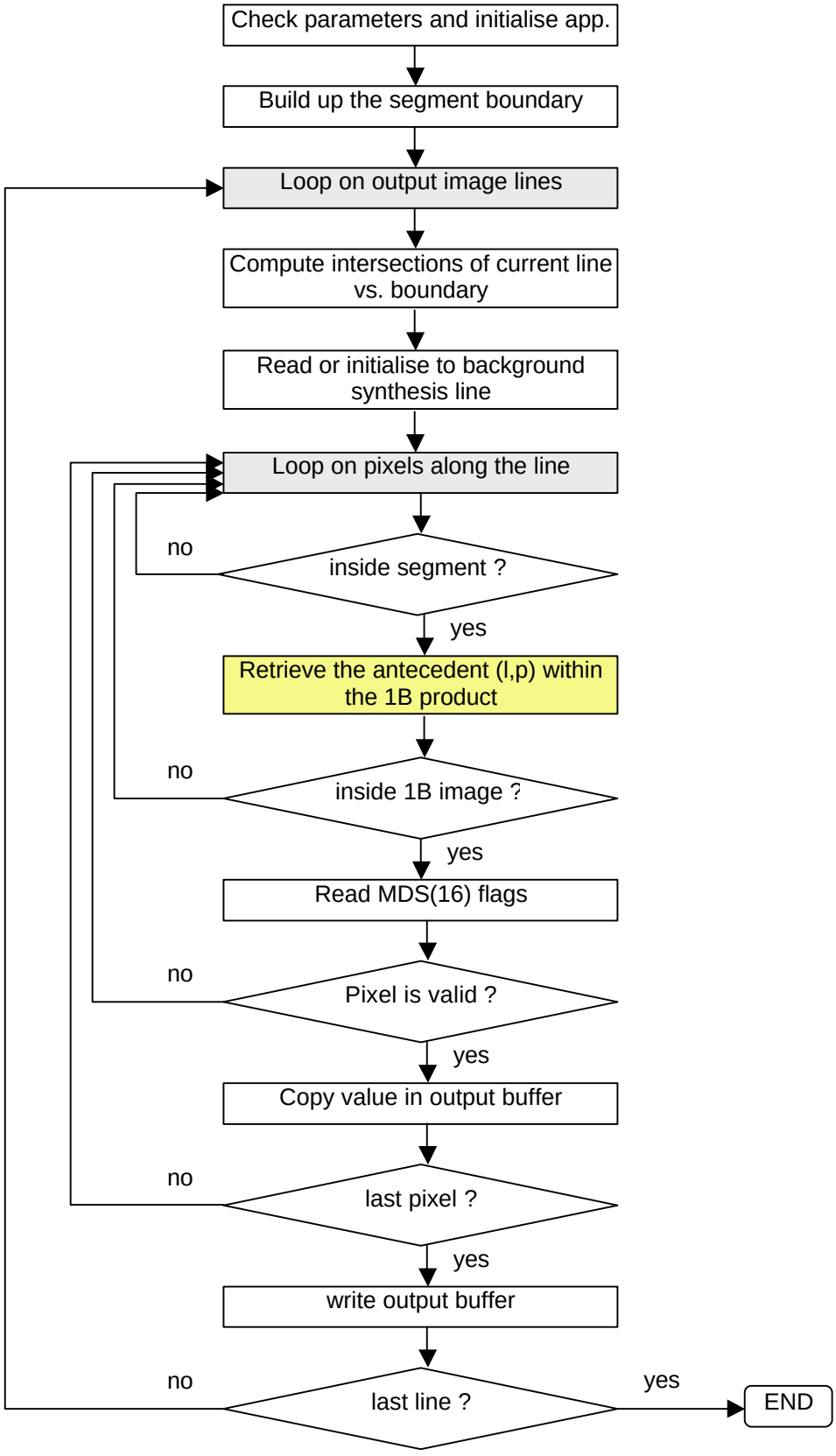


fig. 16 - MERSYN overall workflow

Check parameters and initialise application

MERSYN checks that all the parameters have consistent values according to the three use cases. For example if a synthesis is provided in input (use cases 1 or 2), the number of channels, the projection, origin and pixel size are automatically found in the TELIMAGO header of the input synthesis and therefore these parameters shall not be provided.

MERSYN also initialise the files, reading input headers or decoding the MPH and SPH of the MERIS product in input, allocate the buffers, initialise the variables…

To minimise file access, height of input MERIS buffers shall be large enough to avoid multiple readings when processing all the pixels of one output line. Buffer height shall be greater than the tangent of the maximum NADIR inclination multiplied by the segment width.

Build up the segment boundary

The segment boundary is a closed polygon built from the geodetic information (latitude and longitude) of the tie-points located along the border of the tie-point grid. Figure fig. 17 here attached illustrates an example of MERIS RR segment boundary.

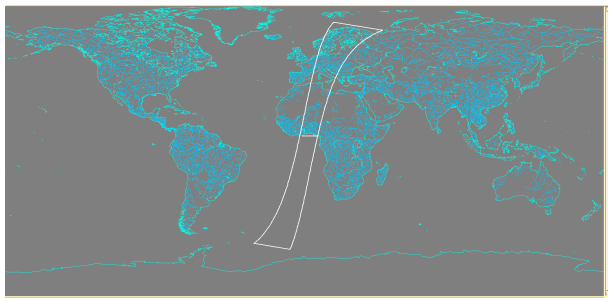


fig. 17 - Segment boundary of a MERIS RR segment.

The segment boundary is used in two ways:

- Optimisation – to compute a projected product, each pixel of the output image shall be processed looking for its antecedent within the input image (see the "reverse location model" hereafter). This process is very time-consuming.

  Identifying if a pixel is inside or not the segment boundary enables processing the strict minimum of output pixels. In fig. 18, only the pixels (represented in orange) located inside the segment boundary (red) among all the pixels of the full line (yellow) shall be processed.

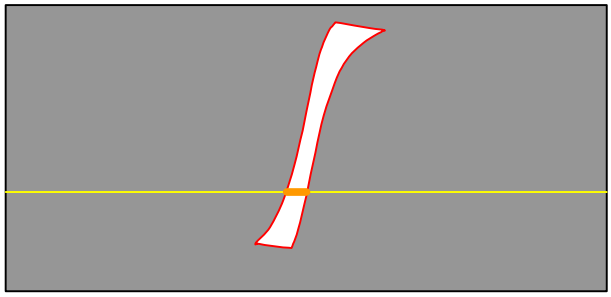


fig. 18 - Processing only pixels inside boundary.

- Output image size – in use case 3, when no synthesis is provided in input and MERSYN is used as a mapping process, the size of the output image shall be determined.

  The retained size is the smallest rectangle that includes all the segment boundary (see fig. 19).

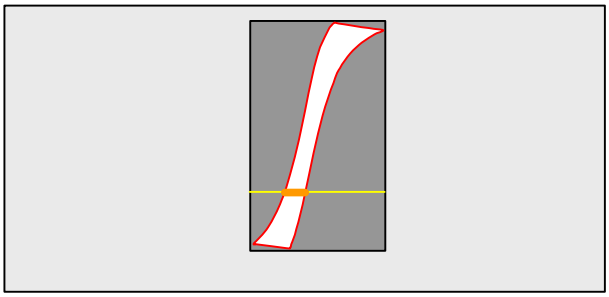


fig. 19 - Computing the output image size.

When the segment is crossing the –180° or +180° meridians (see fig. 20) and if the output projection is bounded to these meridians (case of Mercator, Plate Carrée…), the polygon shall be split in two polygons.

A generalised approach should be to enable splitting in more than two polygons in order to comply with discontinuous coordinates reference systems like the Goode-Homolosine. This approach is not supported in the current version 2.03 of MERSYN.

fig. 20 - Segment crossing the –180°/+180° meridian.

Compute intersections of current line vs. boundary

This operation produces a series of column coordinates delimiting the segment(s) to be processed (represented in orange in fig. 18, fig. 19 or fig. 20) along the current line.

The case of only one point matching one boundary extremity shall be avoid.

Read or initialise to background synthesis line

When a synthesis is provided in input (use cases 1 or 2), the current line i is read and is copied into the output buffer. Otherwise (use case 3), the buffer of the output line is initialised with the background value meaning what no antecedent has been yet found for the output pixels.

Retrieve the antecedent (l,p) within the 1B product

The prediction/correction algorithm used to retrieve the antecedent (l,p) in level 1B image of output pixel (i,j) is fully described in section 3.2 below.

In the current version of MERSYN, only the “nearest neighbour” sampling method has been implemented, and therefore only one pixel (l,p) radiance will be copied into the output buffer. More sophisticated sampling methods could be implemented like the bi-linear interpolation requiring 2x2 pixels anchored on (l,p) or bi-cubic interpolation requiring 4x4 pixels around (l,p).

Read MDS(16) flags and check pixel validity

The flag values matching input pixel (l,p) are read. Except for use case 3, where all the pixels shall be reported in output image, a pixel is considered as being valid if the “glint risk” (TBC), “bright” (TBC) and “invalid” flags are not set.

3.2 prediction/correction algorithm

To produce a geocoded image (see fig. 21), one must handle:

- a direct location model f(l,p,h) able to compute the (λ,φ) coordinates in the Earth reference system associated to the geocoded image (i.e. the projection of the image),
- an inverse location model $f^{-1}(\lambda,\varphi,h)$ able to compute the coordinates (l,p) in the input product reference system (here a MERIS 1B product of m frames and n columns).

Both functions are useful. The direct location model enables computing the coordinates of the image boundary and therefore the size of the output image. The inverse location model forms the kernel of the geocoding processing, because its enables retrieving the antecedent of any point of the output image.

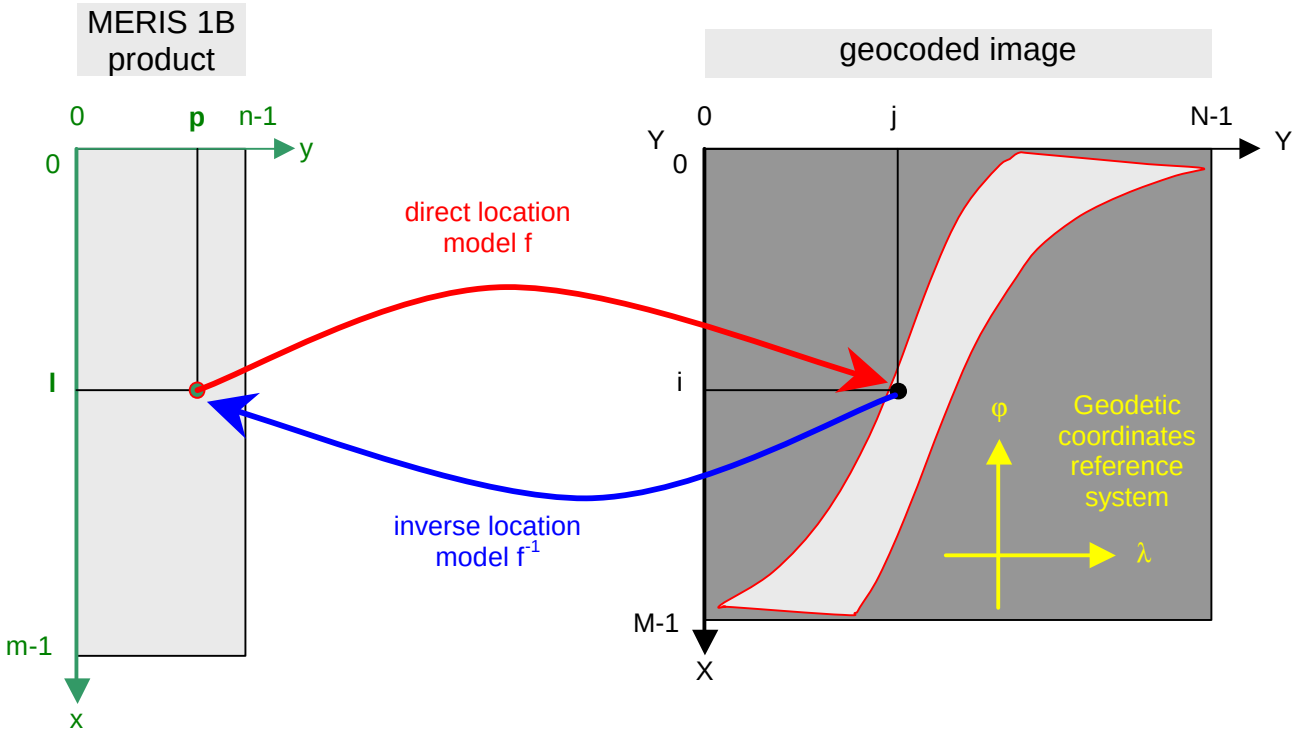


fig. 21 - Direct and inverse location models.

The basic geocoding program is given here below:

```
for i←0 to M-1 do
        for j←0 to N-1 do
                (λ,φ) ← F_proj(i,j)
                (l,p)  ← f^-1(λ,φ,h)
                if (l,p) is inside the input image
                        S(i,j) ← R(l,p)
                fsi
        done
done
```

Where:

F_{pro} is the function giving the Earth coordinates according to the location (i,j) in the output image.

For Cartesian projections system, this function is a simple linear formula:

$$\begin{cases} l = origin_Y + pixel\_width \times j \\ j = origin_X - pixel\_height \times i \end{cases}$$

Prediction/correction principle

The prediction/correction algorithm is to be used when the direct and inverse location models have not the same accuracy. In this case, go and return to/from the output image does not enable retrieving the origin pixel.

$$f\,(\,f^{-1}\,) \neq Id \qquad \text{(the identical function)}$$

In most of the cases the direct location model is issued from a viewing model given by analytical functions and providing accurate results, while the inverse location model is only a predictor, for example estimated by a polynomial function.

In our case, the direct location model makes use of the tie-point geodetic coordinates that are supposed to have been computed with high accuracy.

Scope of the prediction/correction algorithm is to retrieve the pixel (l,p) that match the (λ,φ) by the direct location model: f(l,p) = (λ,φ).

Principle is to sum a series of always-smaller corrections vectors, each one being estimated by a go and return from/to the MERIS segment to/from the geocoded image. The refinement loop is stopped when the (l_i,p_i) is close enough the first point (l_0,p_0) according to a predefined tolerance.

Figure fig. 22 illustrates the prediction/correction principle on two iterations.

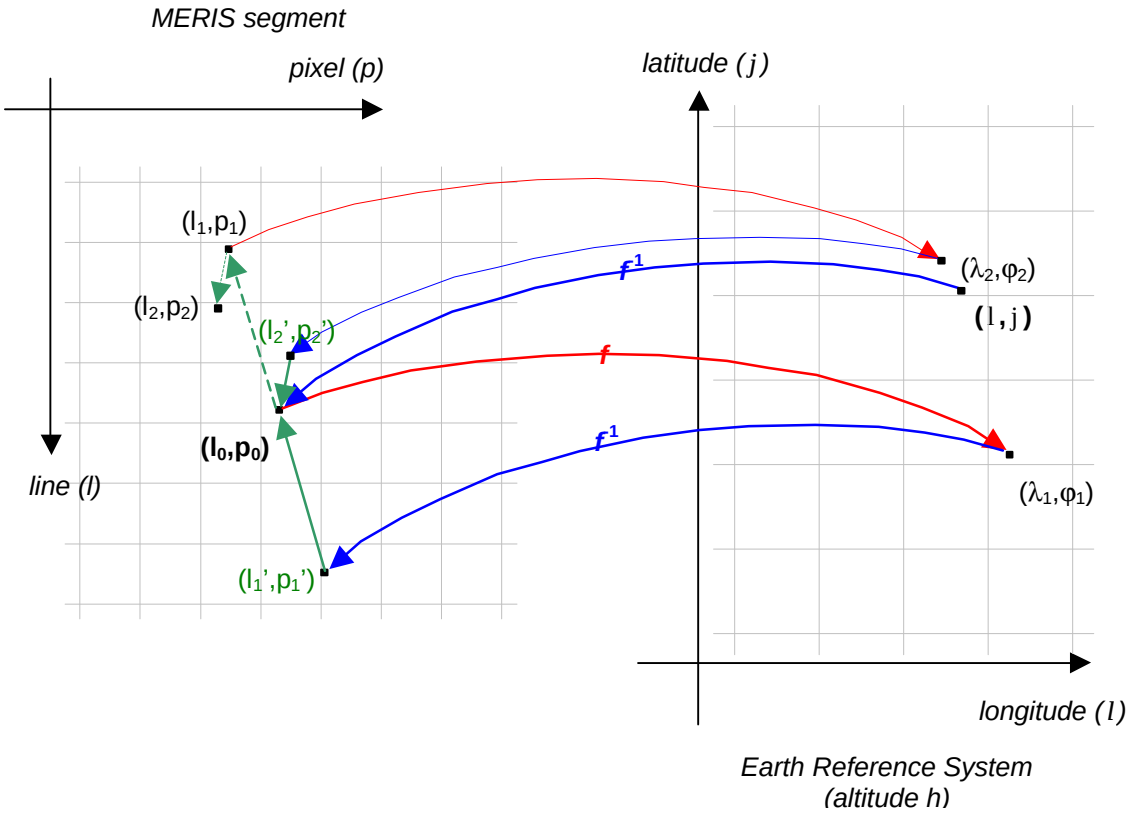


fig. 22 - Example of prediction/correction refinements.

The prediction/correction algorithm is given in fig. 23 here below.

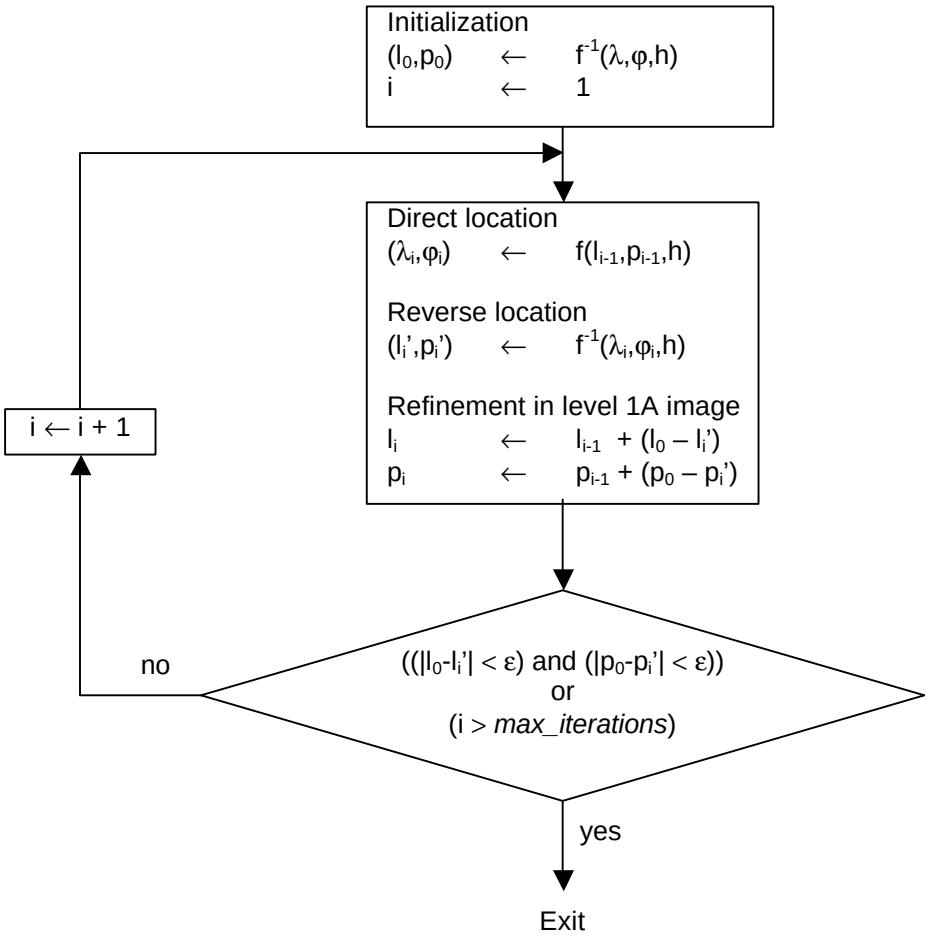


fig. 23 - prediction/correction algorithm.

3.3 Direct location model

The direct location f compute the geodetic coordinates (λ,φ) from the MERIS segment coordinates (l,p) and the altitude h. The provided geodetic coordinates are corrected of the parallax bias taking into account the viewing angle (θ_v,φ_v) and the altitude h.

Retrieving tie-point facet

Coordinates (F,J) of the upper-left tie-point of the facet containing the point (l,p) may simply be obtained dividing l and p by the number DF of frames between two consecutive tie-points and the number DJ of columns between tie-points respectively.

$$\begin{cases} F = \text{int}\left(\frac{l}{DF}\right) \\ J = \text{int}\left(\frac{p}{DJ}\right) \end{cases} \qquad \text{Eq. 3a}$$

$$\begin{cases} f &= l - F \times DF \\ j &= p - J \times DJ \end{cases}$$ Eq. 3b

Geodetic coordinates interpolation

The standard bi-linear interpolation (eq. 1) is applied to the longitude and latitude of the tie-points.

$$\begin{aligned} l(l,p) &= \left(\frac{DJ-j}{DJ}\right) \times \left[\left(\frac{DF-f}{DF}\right) \times Lon(F,J) + \left(\frac{f}{DF}\right) \times Lon(F+DF,J)\right] \\ &+ \left(\frac{j}{DJ}\right) \times \left[\left(\frac{DF-f}{DF}\right) \times Lon(F,J+DJ) + \left(\frac{f}{DF}\right) \times Lon(F+DF,J+DJ)\right] \end{aligned}$$ Eq. 4a

$$\begin{aligned} j(l,p) &= \left(\frac{DJ-j}{DJ}\right) \times \left[\left(\frac{DF-f}{DF}\right) \times Lat(F,J) + \left(\frac{f}{DF}\right) \times Lat(F+DF,J)\right] \\ &+ \left(\frac{j}{DJ}\right) \times \left[\left(\frac{DF-f}{DF}\right) \times Lat(F,J+DJ) + \left(\frac{f}{DF}\right) \times Lat(F+DF,J+DJ)\right] \end{aligned}$$ Eq. 4b

Viewing angle interpolation

The standard bi-linear interpolation (eq. 1) is applied to the viewing zenith and viewing azimuth of the tie-points.

$$\begin{aligned} q_v(l,p) &= \left(\frac{DJ-j}{DJ}\right) \times \left[\left(\frac{DF-f}{DF}\right) \times q_v(F,J) + \left(\frac{f}{DF}\right) \times q_v(F+DF,J)\right] \\ &+ \left(\frac{j}{DJ}\right) \times \left[\left(\frac{DF-f}{DF}\right) \times q_v(F,J+DJ) + \left(\frac{f}{DF}\right) \times q_v(F+DF,J+DJ)\right] \end{aligned}$$ Eq. 5a

$$\begin{aligned} j_v(l,p) &= \left(\frac{DJ-j}{DJ}\right) \times \left[\left(\frac{DF-f}{DF}\right) \times j_v(F,J) + \left(\frac{f}{DF}\right) \times j_v(F+DF,J)\right] \\ &+ \left(\frac{j}{DJ}\right) \times \left[\left(\frac{DF-f}{DF}\right) \times j_v(F,J+DJ) + \left(\frac{f}{DF}\right) \times j_v(F+DF,J+DJ)\right] \end{aligned}$$ Eq. 5b

Compute and apply the geodetic correction

The geodetic correction formula applied to the tie-points (see eq. 2 and fig. 10 in section "Parallax estimate at tie-points" of section 2.2) are applied to the (l,p) point.

$$\begin{aligned} dx &= h \times \tan(q_v(l,p)) \\ dlat(l,p) &= \frac{dx \times \cos(j_v(l,p))}{R_e} \\ dlon(l,p) &= \frac{dx \times \sin(j_v(l,p))}{R_e \times \cos(j(l,p))} \end{aligned}$$ Eq. 6a

$$\begin{aligned} l(l,p) &\leftarrow l(l,p) + dlon(l,p) \\ j(l,p) &\leftarrow j(l,p) + dlat(l,p) \end{aligned}$$ Eq. 6b

3.4 Inverse location model

The inverse location f^{-1} retrieves the facet (F,J) that includes the point which geodetic coordinates are (λ,φ) and computes the offset (f,j) within the facet.

Retrieving the tie-point facet

Facet retrieving is one of the most time-consumer, and therefore an optimisation is proposed assuming that processing of geocoded image is performed on contiguous points: … (i,j) (i,j+1)… (i,N-1) (i+1,1)…

Coordinates (F_{last},J_{last}) of the latest facet are kept and searching is performed in this facet first (see fig. 24). In case of failure, searching is performed within the four nearest (4-connex) facets, and then within the 4 other neighbour (8-connex) facets. In case of failure, all the facets of the tie-point grid shall be inspected.

This last case is significantly time-consuming (in particular for RR products containing a large number of tie-points) and all shall be done to avoid it.

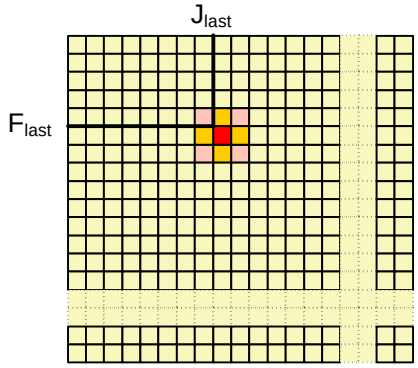


fig. 24 - Retrieving tie-points facet order.

Retrieving offsets within the facet

Facets are obviously not parallel to the geodetic reference system and placed in this referential (see fig. 25), they may have whatever pattern.

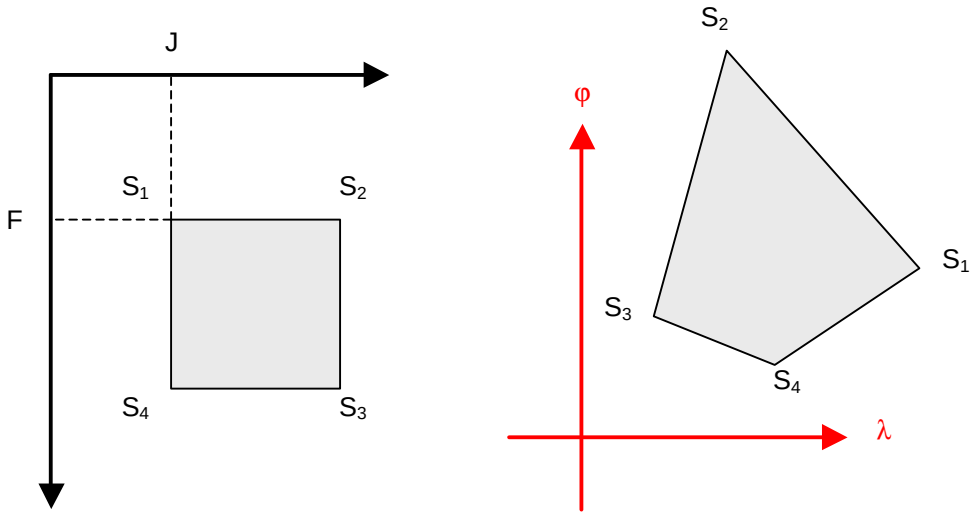


fig. 25 - Facet in the level 1B grid referential (left) and in the geodetic referential (right).

To state if a point is within a facet or not, we cannot only test that the geodetic coordinates (λ,φ) of the point are within the longitude and latitude ranges of the tie-point corners. Figure fig. 26 here below for example illustrates an example where the latitude and longitude of a point are inside the ranges defined by the tie-points but the point is outside the facet.

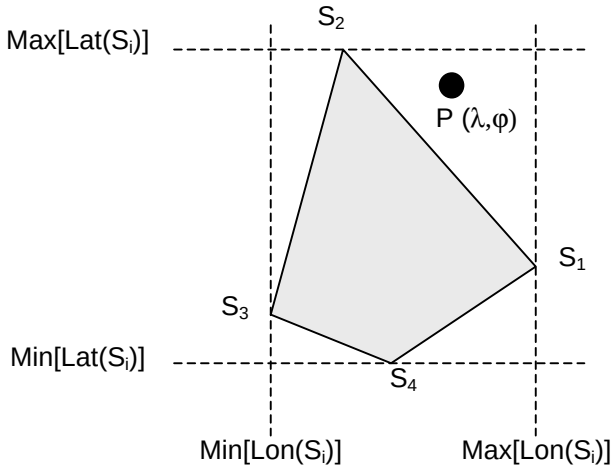


fig. 26 - Point inside the latitude and longitude ranges but outside the facet.

Algorithms exists to determine if a point is inside a polygon or not but they are time-consuming.

It is proposed to use a method illustrated in fig. 27 that:

- computes the two intersection points I_1 and I_2 of the facet opposite borders (when not parallel),
- computes the intersection I'_1 and I''_1 of (I_1P) with the two other borders,
- computes the intersection I'_2 and I''_2 of (I_2P) with the two other borders,
- checks that points I'_1, I''_1, I'_2 and I''_2 are inside the segments [S_1S_4], [S_2S_3], [S_1S_2] and [S_3S_4] respectively.

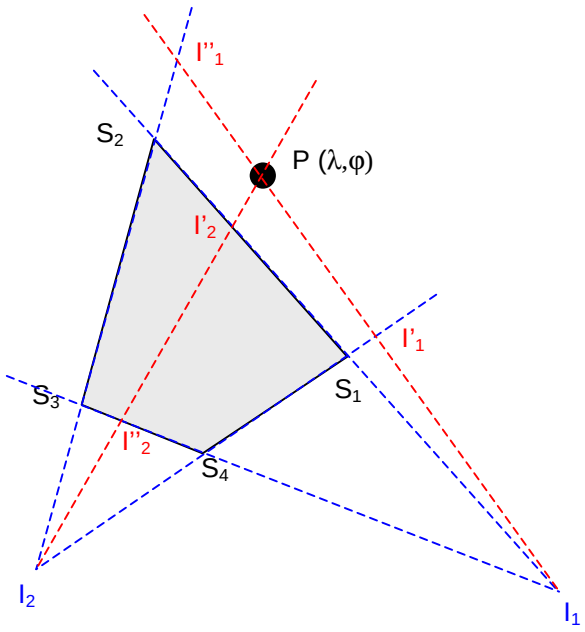


fig. 27 - Intersections method.

4 ORTHORECTIFICATION RESULTS

Images given in this section have been generated by the MERSYN process, setting different parameters like the DEM in input.

Geometry checking is performed comparing the orthorectified MERIS image mapped in “Plate Carrée” projection with ground spacing of 260 metres (FR) with reference to data like:

- Vector layers and in particular those showing the coastlines.
- Landsat TM mosaic that has been resampled to 260 metres in the same projection as for the MERIS images.

In section 4.7, a non-orthorectified scene produced by BEAM / VISAT will be compared with the same MERSYN result.

4.1 MERIS scene in input

The full resolution (FR) MERIS scene being controlled has the standard name

MER_FR__1PNUPA20030921_092341_000000982020_00079_08149_0534.N1

The geometry of the level 1B product is shown in fig. 28.

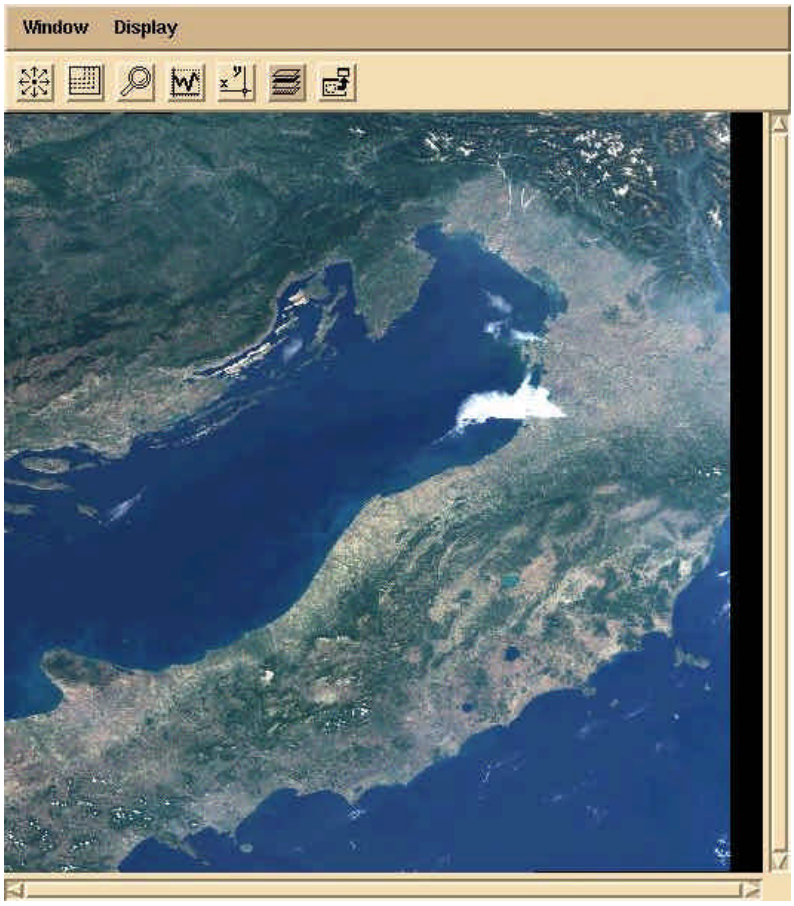


fig. 28 - Level 1B MERIS scene.

This scene has been chosen because its footprint (see fig. 29) includes terrains with very varying elevations (Alps, plain of Pô, Abbruzes…).

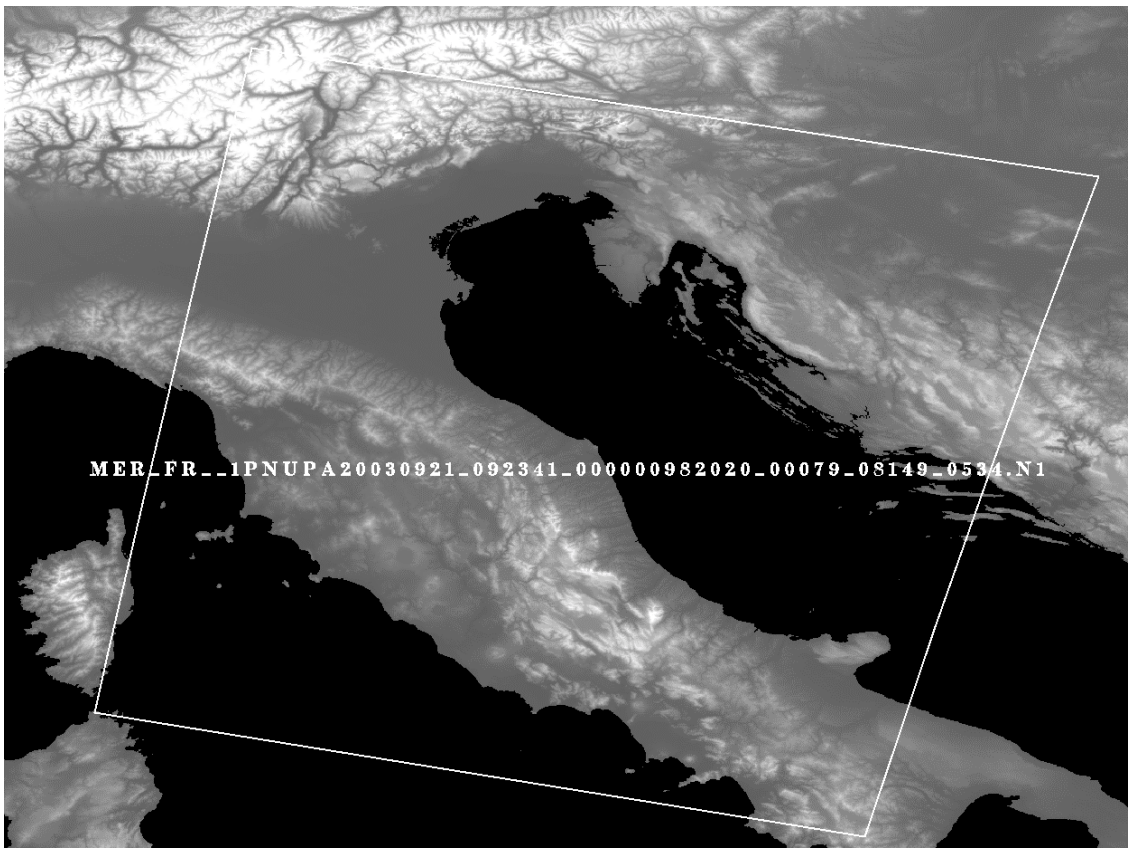


fig. 29 - MERIS footprint overlaid on SRTM elevations

In particular, Alps located at the upper-left corner of the scene are acquired with a high zenith angle. Some of the values found within the "Tie-point ADS" are given in section A.1 (Decoding log) of APPENDIX A.

```
DATA SET #3 - "Tie points ADS":
 No.         Date
FIRST  (Lat., Lon.)       Alt. Rough.  Corr. Lat. Lon.      Sun (Zen., Azi.)  View (Zen., Azi.)
LAST   (Lat., Lon.)       Alt. Rough.  Corr. Lat. Lon.      Sun (Zen., Azi.)  View (Zen., Azi.)

  1  117451421.714409
( 45.909863,  18.014693 )    98    3      0 1e-6       0 1e-6 ( 48.2, 153.6 )  (  0.0, 182.0 )
( 47.028196,  10.603496 )  1476  382  -1907 1e-6  -16512 1e-6 ( 51.8, 145.0 )  ( 40.7,  99.6 )

  9  117451444.240873
( 44.594348,  17.519381 )   591  159      0 1e-6       0 1e-6 ( 47.2, 152.5 )  (  0.0, 181.9 )
( 45.703992,  10.278711 )   850  227  -1101 1e-6   -9284 1e-6 ( 50.8, 144.1 )  ( 40.7,  99.6 )

 17  117451466.767337
( 43.277165,  17.041762 )   417   32      0 1e-6       0 1e-6 ( 46.1, 151.4 )  (  0.0, 181.9 )
( 44.379209,   9.960598 )   744   75   -967 1e-6   -7942 1e-6 ( 49.8, 143.2 )  ( 40.7,  99.7 )
```

Zenith of the viewing angle of the first CCD (matching the East of the scene) is almost 0.0 meaning that pixels are located at the NADIR of the satellite, while the viewing angle of the last CCD (West of the scene) has almost the maximum value (40.7°).

The MERIS FR scene is located on the East of the track leading to the left border region, for example North of the Alps, is seen with the maximum incidence angle and therefore showing the largest parallax defect.

4.2 Digital elevation models

Two digital elevation models have been used to orthorectify the MERIS scene. These two DEMs have not the same resolution to enable comparing the influence of this feature.

DEM Italy

This 250 metres DEM has been generated by photogrammetry techniques. This DEM (see fig. 30) covers the administrative boundaries of Italy and the North of Alps.

fig. 30 - DEM Italy (250 metres)

SRTM

SRTM30 is a near-global digital elevation model (DEM) comprising a combination of data from the Shuttle Radar Topography Mission, flown on February 2000 and the U.S. Geological Survey's GTOPO30 data set. It can be considered to be either an SRTM data set enhanced with GTOPO30, or as an upgrade to GTOPO30.

SRTM is provided by tiles and its ground sampling is 30'' arc (approximately 1 km resolution at Equator). The tile covering the MERIS scene (see fig. 31) is identified **W020N90** (geodetic coordinates of the upper left corner) and as an extent of 40° (West - East) by 50° (North - South).

fig. 31 - SRTM W020N90 tile

4.3 Reference image – The Landsat TM mosaicsna

The reference image used to perform controls is a mosaic of Landsat TM scenes with a resolution of 30 metres.

This reference scene has been computed by GAEL Consultant in September 1997 from 38 Landsat scenes and orthorectified using a 75 metres DEM of Italy.

Ground control points have been taken from 1:20.000 and 1:25.000 IGM maps.

The location accuracy has been controlled with a RMS error less than 50 metres.

Figure fig. 32 shows the full image (top) and a 1:100.000 display (bottom) overlaid by vector layers of the Italian provinces.

Note: Sites are now available to download orthorectified Landsat ETM+ scenes (see http://image2000.jrc.it/ or http://glcf.umiacs.umd.edu/index.shtml).

fig. 32 - Landsat TM mosaic

4.4 Superimposability – Visual checking

A visual verification of the superimposability quality may be performed using the "Flicker" tool of REGIST application (see fig. 33). In this example, the MERIS projected image (left) is compared with the Landsat mosaic (right).

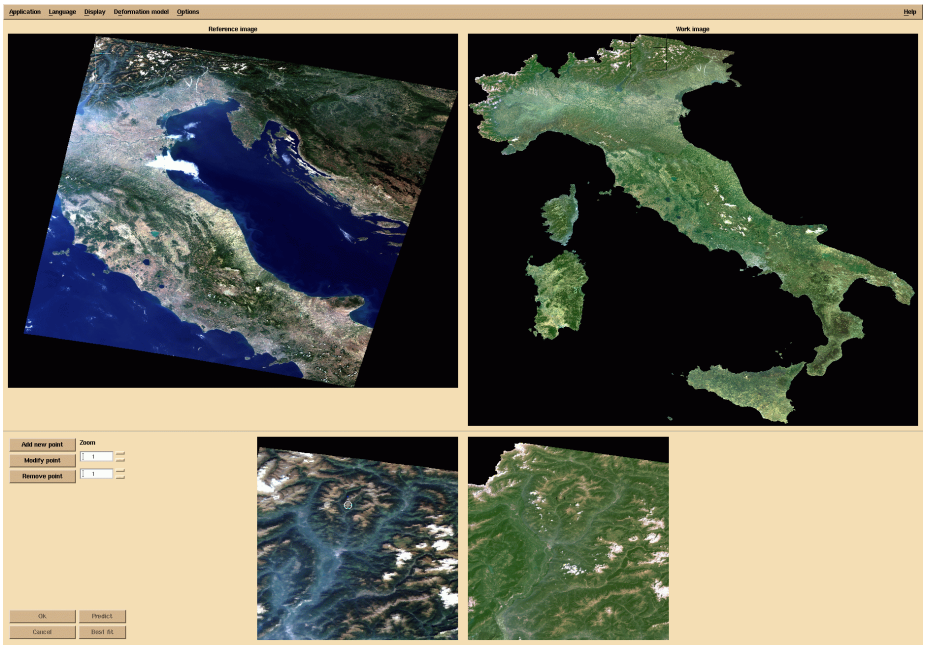


fig. 33 - REGIST application (snap01_REGIST.gif)

For the selected zone, full resolution or zoomed windows found in MERIS and Landsat are alternatively displayed within a same frame (see fig. 34 and fig. 35).

fig. 34 - MERIS (snap02_REGIST_Flicker_no-ortho_meris.gif)

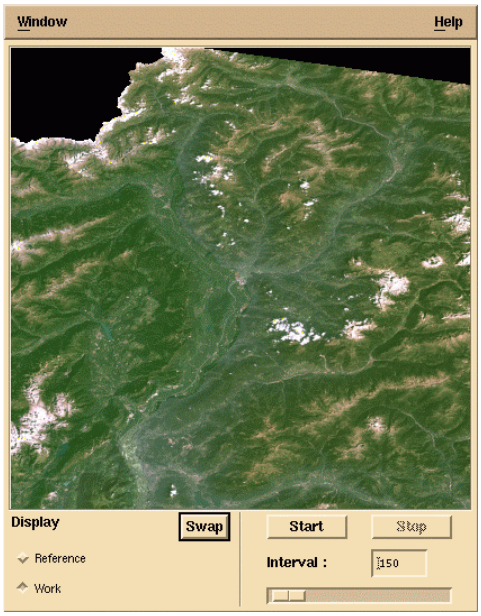


fig. 35 - Landsat (snap02_REGIST_Flicker_no-ortho_tm.gif)

This alternative displaying (flicking) enables identifying sub-pixel displacements. Such movements cannot be observed in this document, but GIF images have been captured and stored in GIF animations that are available on demand:

- snap02_REGIST_Flicker_no-ortho_anim.gif
- snap03_REGIST_Flicker_alti-grid_anim.gif
- snap04_REGIST_Flicker_dem-italy_anim.gif
- snap05_REGIST_Flicker_dem-italy+translation_anim.gif
- snap06_REGIST_Flicker_dem-srtm+translation_anim.gif

Comment of animation 02 – No orthorectification

When no orthorectification is performed the deformations appear with the largest amplitude.

Comment of animation 03 – Altitudes from tie-points grid

When orthorectification is computed interpolating the elevation values from those found within the tie-point grid, we may observe that the bottom of the valleys are almost correctly superimposed (no movement) but the mountains are moving eastward.

Such an internal geometry defect is due to a poor accuracy of the tie-point elevations that have been computed from a 5' arc (almost 9 km along equator) DEM smoothing the picks and ridges of the mountains.

For example a point (coordinates (117,872) in the geocoded scene) has the following elevation value depending on the source, given here in increasing order of spatial accuracy:

Source	resolution	altitude
Tie-point grid	5' arc	**1406**.003360 m
DEM SRTM	30''	**2106**.815331 m
DEM Italy	250 m	**2230**.704269 m

The deficit of 700 metres (2106 – 1406) on pixels seen with the largest viewing zenith (40°) leads to an error of approximately 587 metres (= tan(40°) x 700 m, see eq. 2), i.e. more than two (2) pixels.

While the difference between the SRTM DEM and the accurate one (250 m) over Italy shows a deficit of 124 metres leading to a parallax error of only 104 metres, i.e. a third of the MERIS RR resolution.

Figure fig. 36 here below illustrates the parallaxes error that can be due to the accuracy of the Digital Elevation Model.

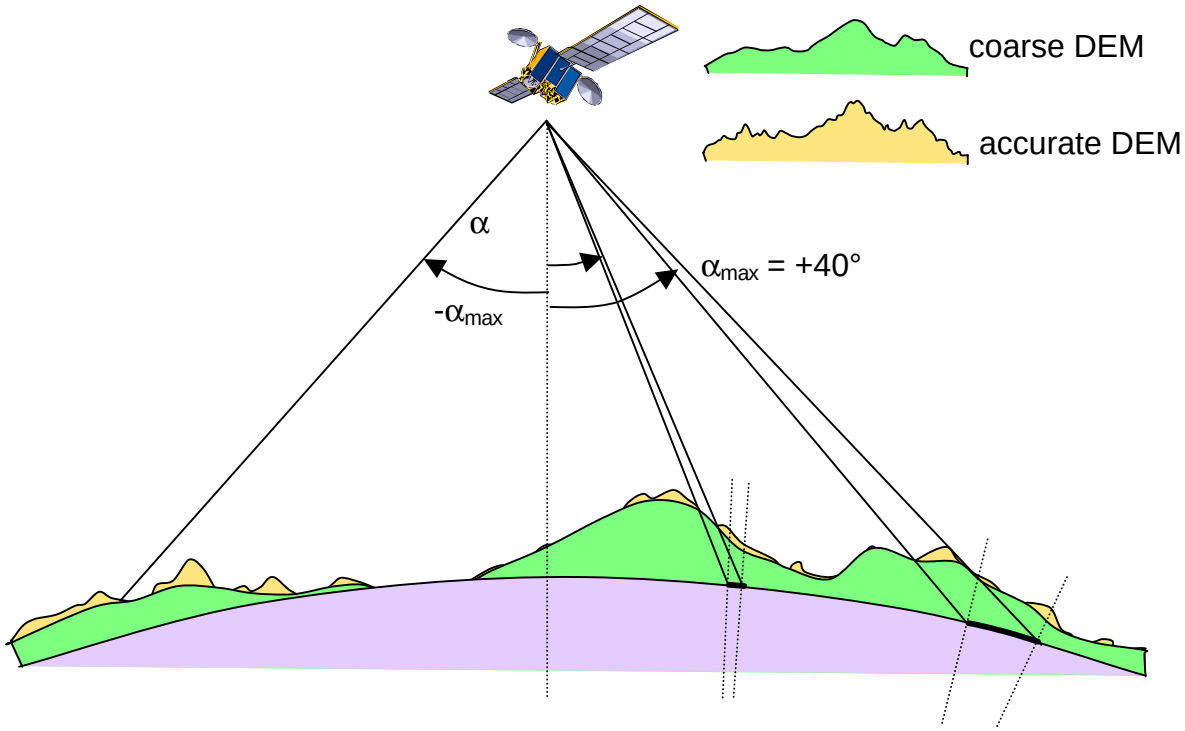


fig. 36 - Parallaxes error due to the precision of the DEM(s)

Comment of animation 04 and 05 – Altitudes from DEM Italy

When orthorectification is performed from elevation values found in DEM Italy (250 m planimetric resolution), the two images (see animation 04) are shifted in a uniform way.

Such translation has an amplitude of +1.2 pixels vertically and –4.4 pixels horizontally, leading to an error modulus of **1185 metres**.

"Moving" the orthorectified MERIS scene (i.e. just changing the origin coordinates of the upper left corner), the MERIS and Landsat scenes are perfectly superimposable (see animation 05).

Comment of animation 06 – Altitudes from DEM SRTM

Exactly the same translation defect has been observed in the MERIS scene orthorectified from elevations found within the SRTM DEM (30 '' arc planimetric resolution).

"Moving" the orthorectified MERIS scene with the same vector (+1.2,-4.4) leads to the same good result, superimposable on the Landsat mosaic.

Conclusions

From the observations detailed here above we may conclude the following three assessments.

- Quality of the tie-point elevations – The quality of the elevations found within the tie-point grid is not sufficient to correct parallax defect below 2 pixels error.
- Uniform translation – For this scene, a large translation of 1185 metres is observed leading to an absolute location accuracy of 4.5 pixels (see fig. 37). This uniform translation has been measured on only one scene and cannot be generalised for any other MERIS RR acquisition. A systematic control of a significant set of MERIS scenes is out of the scope of this handbook.
- Quality of the DEM – SRTM elevation data seems to be enough to efficiently orthorectify MERIS FR scenes. Results obtained are similar to those obtained with a more accurate DEM of 250 metres resolution.

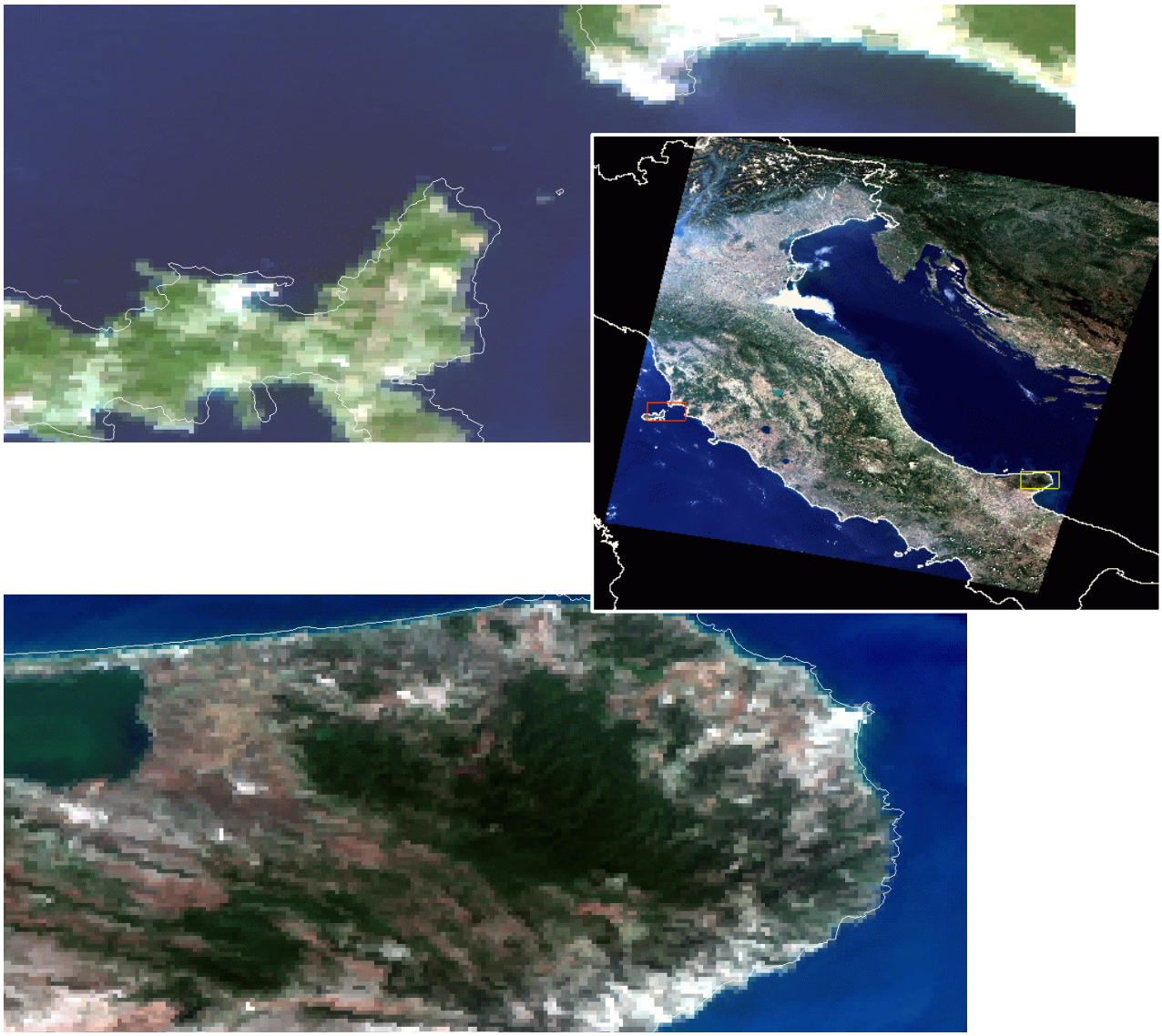


fig. 37 - West offset along the coastline

4.5 Run-time performances

Orthorectification is time consuming. Most of the time is spent to access elevation data from an external DEM, retrieve the tie-point facet including the processed point, go and return according to the prediction/correction algorithm, projection changes…

Table here below resumes the run-time and convergence performances observed for the generation of the four synthesis.

	No orthorectification	Altitudes from tie-points grid	Altitudes from DEM Italy	Altitudes from DEM SRTM
Run-time	**27.700 s**	**68.760 s**	**325.960 s**	**304.440 s**
Iteration 0	7818186 (100.0 %)	2936 (0.0 %)	2936 (0.0 %)	2936 (0.0 %)
Iteration 1	-	2852732 (36.5 %)	3008252 (38.5 %)	3034335 (38.8 %)
Iteration 2	-	4833499 (61.8 %)	4798944 (61.4 %)	4777361 (61.1 %)
Iteration 3	-	129019 (1.7 %)	8054 (0.1 %)	3554 (0.0 %)

table 3 - Run-time performances.

Performance of the prediction/correction algorithm

Performance of the prediction/correction algorithm almost doubles the execution time (68.760 seconds vs. 27.700 seconds).

We may note that only three iterations are required to converge on a precision that was set to 0.1 pixels (i.e. 26 metres maximum error between the direct and inverse location models).

Access time to DEMs

Accessing to external data almost multiply by five (x5) the execution time (325.960 seconds or 304.440 seconds vs. 68.760 seconds).

Note that a bi-linear interpolation of the DEM value being computed, a 2x2 window of elevation values shall be read for each pixel to be processed. Access to the elevation data is performed inside the prediction/correction loop.

Even if sizes of the DEMs are very close (5800 lines x 4400 pixels for the DEM Italy and 6000 lines x 4800 pixels for the DEM SRTM), the time to access DEM Italy is a little larger that the one to access DEM SRTM. This extra time is only due to the precision of the DEM (250 metres for Italy vs. 30'' arc for SRTM), inducing more frequent accesses to the DEM Italy file.

4.6 Disparity Analysis

Misregistration of MERIS scene with regard to the Landsat TM mosaic may also be computed performing a disparity analysis (DISPAR process).

Research of homologous points is accomplished within an exploration window of 9 lines x 19 columns. Pattern matching is performed using the classical ZNCC (Zero mean Normalized Cross Correlation) algorithm, computing the normalised covariance within statistical window which size is 5x5.

DISPAR has been applied to the synthesis for which no orthorectification has been applied (file "synthesis_no-ortho.sta.3" and to the synthesis Orthorectified from the SRTM DEM (file "synthesis_dem-srtm.sta.3").

Obtained results are given in the logs here below. Figures fig. 38 and fig. 39 show the points or regions with the highest confidence value, the minimum threshold having been set to 95%. The vectors illustrate

the tie-point automatically generated with a pixel sampling of 0.1 pixels and greater than the 95% confidence threshold.

Better correlation of SRTM DEM synthesis

The regions showing the much variation of landscape like the coastlines, the lake borders and the mountains are those in which the correlation is higher.

As expected, the better internal geometry of the orthorectified synthesis leads to more points (58 "CANDIDATE PIXELS" (*)) having a confidence index over 95% than for the non-orthorectified image (only 37 points).

Nevertheless, both scores are surprisingly low, meaning that the radiometric correlation between the band 3 of MERIS and band 1 of Landsat is poor even if both bands are cantered on the blue of the visible spectrum. The same defect has been observed in the green bands (5 for MERIS and 2 for Landsat) or the red bands (7 for MERIS and 3 for Landsat).

The reasons of such a decorrelation (that could be due to the precision of the radiometric bands, their bandwidth, the different acquisition periods…) should be investigated but such an analysis is out of the scope of this document.

Global displacement

Individual displacements (dX,dY) are more scattered for the non-orthorectified scene. This defect is particularly visible within the region of North Alps (see fig. 38). Nevertheless, a global displacement noted in the previous section can be estimated by the algebraic mean of the (dX,dY) displacements:

- (**`-1.029730`**, **`2.813514`**) for the non-orthorectified scene, and
- (**`-0.724138`**, **`2.591379`**) for the scene Orthorectified from the SRTM DEM.

This last displacement has a modulus of **699.569 metres**.

FILE synthesis_no-ortho.sta.3

```
STATISTICAL WINDOW SIZE   = 5
EXPLORATION WINDOW HEIGHT = 9
EXPLORATION WINDOW WIDTH  = 19
CONFIDENCE THRESHOLD      = 95.0 %

TOTAL NUMBER OF PIXELS =     110544
IMAGE PIXELS           =      74771
CANDIDATE PIXELS  (*)  =         37   (0.0 %)

TIE-POINTS             =         37


CONFIDENCE VALUES:
MINIMUM                =    0.000000 vs.   95.009510 (*)
MAXIMUM                =   99.411765 vs.   99.411765 (*)
MEAN                   =   26.599705 vs.   96.399719 (*)
VARIANCE               =   10.974342 vs.    0.013694 (*)
STANDARD DEVIATION     =   33.127544 vs.    1.170194 (*)



X DISPLACEMENT:
ON LIMIT               = 6268 / 74771 (8.38%)
RMS                    =    1.800067 vs.    1.510951 (*)
MINIMUM                =   -4.000000 vs.   -3.000000 (*)
MAXIMUM                =    4.000000 vs.    2.000000 (*)
MEAN                   =    0.401206 vs.   -1.029730 (*)
STANDARD DEVIATION     =       1.755 vs        1.106 (*)

ORIGINAL VALUES:
RMS                    =  468.017367 vs.  392.847264 (*)
MINIMUM                =-1040.000000 vs. -780.000000 (*)
MAXIMUM                = 1040.000000 vs.  520.000000 (*)
MEAN                   =  104.313651 vs. -267.729730 (*)
STANDARD DEVIATION     =     456.244 vs      287.489 (*)



Y DISPLACEMENT:
ON LIMIT               = 3490 / 74771 (4.67%)
RMS                    =    3.620483 vs.    4.084744 (*)
MINIMUM                =   -9.000000 vs.   -4.100000 (*)
MAXIMUM                =    9.000000 vs.    8.000000 (*)
MEAN                   =    0.941761 vs.    2.813514 (*)
STANDARD DEVIATION     =       3.496 vs        2.961 (*)

ORIGINAL VALUES:
RMS                    =  941.325648 vs. 1062.033491 (*)
MINIMUM                =-2340.000000 vs.-1066.000000 (*)
MAXIMUM                = 2340.000000 vs. 2080.000000 (*)
MEAN                   =  244.857819 vs.  731.513514 (*)
STANDARD DEVIATION     =     908.922 vs      769.937 (*)
```

FILE synthesis_dem-srtm.sta.3

```
STATISTICAL WINDOW SIZE   = 5
EXPLORATION WINDOW HEIGHT = 9
EXPLORATION WINDOW WIDTH  = 19
CONFIDENCE THRESHOLD      = 95.0 %

TOTAL NUMBER OF PIXELS =     110544
IMAGE PIXELS           =      74723
CANDIDATE PIXELS  (*)  =         58   (0.1 %)

TIE-POINTS             =         58


CONFIDENCE VALUES:
MINIMUM                =    0.000000 vs.   95.003623 (*)
MAXIMUM                =  100.000000 vs.  100.000000 (*)
MEAN                   =   26.689998 vs.   96.212369 (*)
VARIANCE               =   11.050492 vs.    0.016515 (*)
STANDARD DEVIATION     =   33.242281 vs.    1.285098 (*)



X DISPLACEMENT:
ON LIMIT               = 6113 / 74723 (8.18%)
RMS                    =    1.776098 vs.    1.288276 (*)
MINIMUM                =   -4.000000 vs.   -3.000000 (*)
MAXIMUM                =    4.000000 vs.    2.000000 (*)
MEAN                   =    0.424984 vs.   -0.724138 (*)
STANDARD DEVIATION     =       1.725 vs        1.065 (*)

ORIGINAL VALUES:
RMS                    =  461.785466 vs.  334.951772 (*)
MINIMUM                =-1040.000000 vs. -780.000000 (*)
MAXIMUM                = 1040.000000 vs.  520.000000 (*)
MEAN                   =  110.495912 vs. -188.275862 (*)
STANDARD DEVIATION     =     448.371 vs      277.029 (*)



Y DISPLACEMENT:
ON LIMIT               = 3246 / 74723 (4.34%)
RMS                    =    3.484071 vs.    3.703842 (*)
MINIMUM                =   -9.000000 vs.   -7.000000 (*)
MAXIMUM                =    9.000000 vs.    8.000000 (*)
MEAN                   =    0.800090 vs.    2.591379 (*)
STANDARD DEVIATION     =       3.391 vs        2.646 (*)

ORIGINAL VALUES:
RMS                    =  905.858579 vs.  962.999015 (*)
MINIMUM                =-2340.000000 vs.-1820.000000 (*)
MAXIMUM                = 2340.000000 vs. 2080.000000 (*)
MEAN                   =  208.023313 vs.  673.758621 (*)
STANDARD DEVIATION     =     881.650 vs      688.053 (*)
```

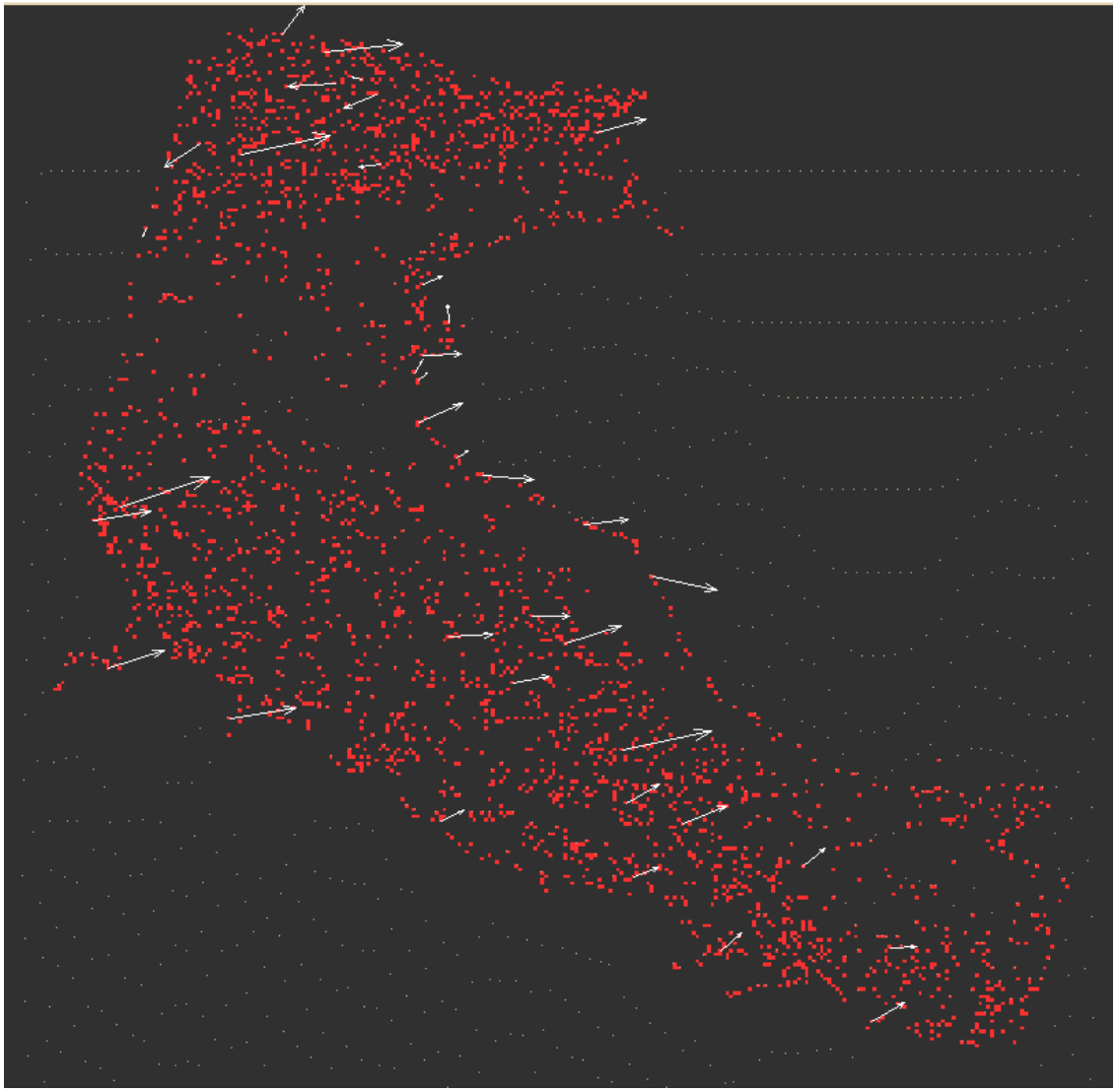
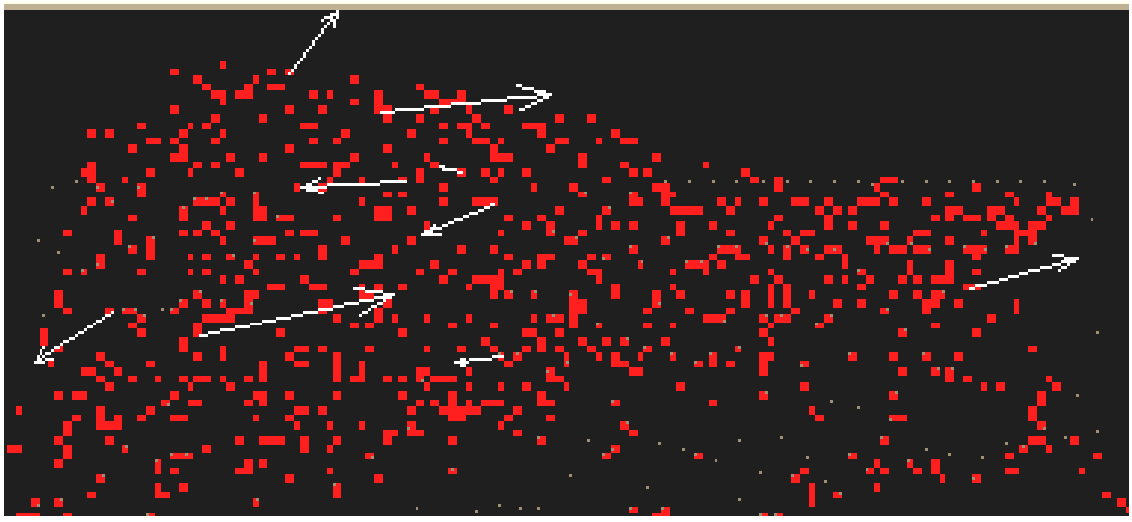


fig. 38 - No-orthorectification – Highest confidence (red) and 95% tie-points



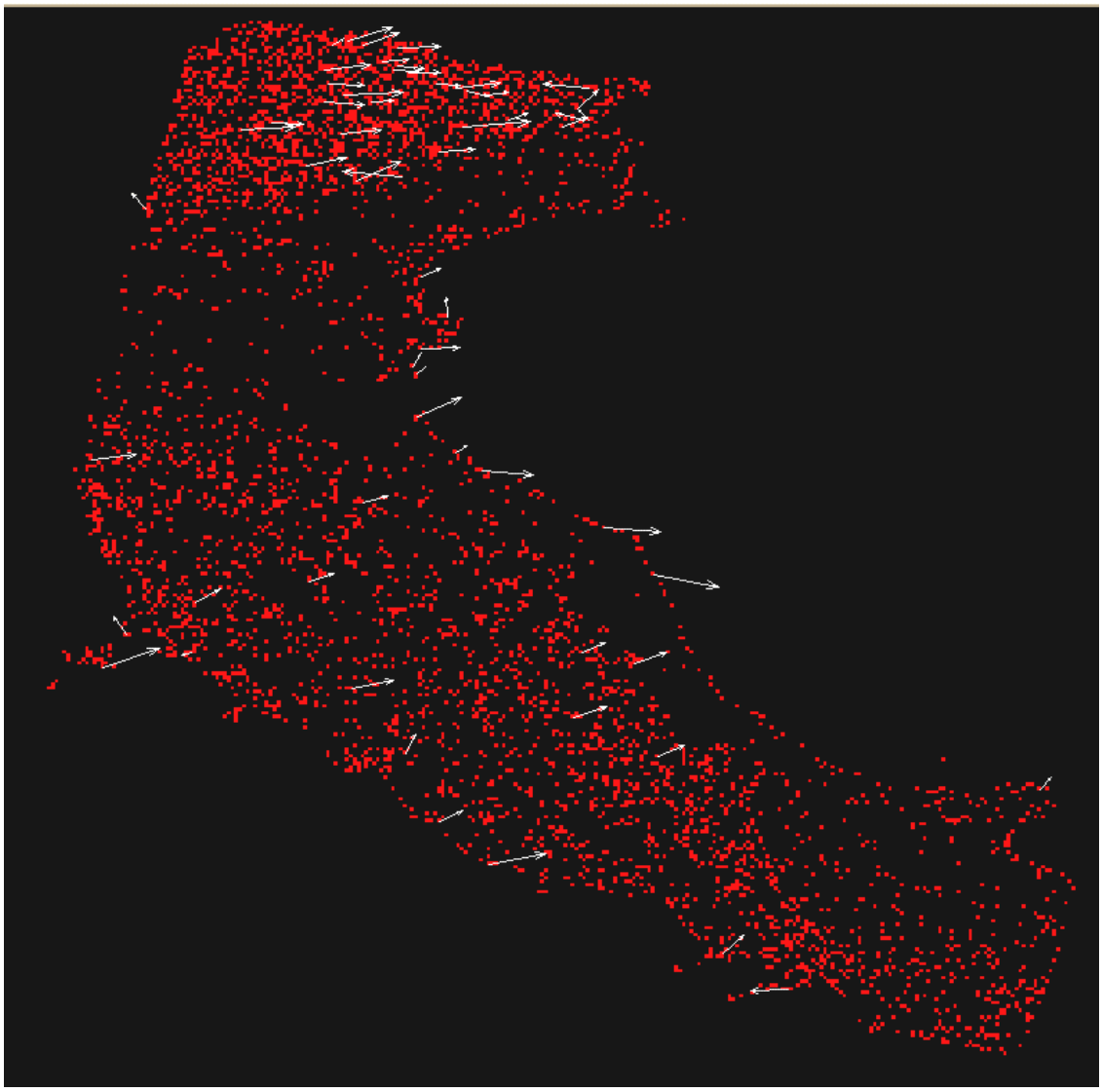
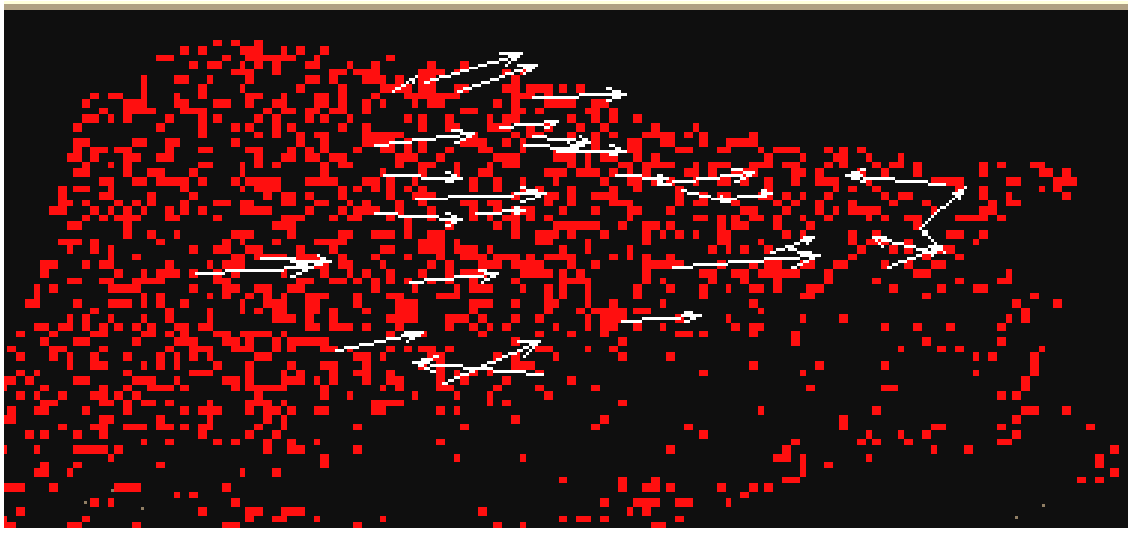


fig. 39 – Orthorectification from SRTM DEM – Highest confidence (red) and 95% tie-points



4.7 Comparison with BEAM / VISAT

Scope of this section is to compare the geometry of the images produced by the BEAM software (downloaded from http://scipc3.scicon.gkss.de/services/beam2/software on 20/08/2004) with the image produced by MERSYN.

In its version 2.3, BEAM / VISAT does not process orthorectification and therefore the image produced by MERSYN is also not orthorectified (parameter "-ort" set to NO).

Both images have been geocoded in the "UTM 31, ellipsoid WGS-84 and Datum WGS-84". Figure here below shows the BEAM image (left) and the MERSYN image (right) with a resolution of 260 metres each one.

An accurate superimposability has been verified setting the Flicker tool with a x4 zoom within the four zones illustrated on the MERSYN image. Behaviour of the REGIST application is described in section **Erreur ! Source du renvoi introuvable.**).

Four animations are available on demand that clearly show a perfect superimposability of BEAM and MERSYN images. The few points that are not exactly superimposed are probably due to the different (nearest neighbour) interpolation methods.

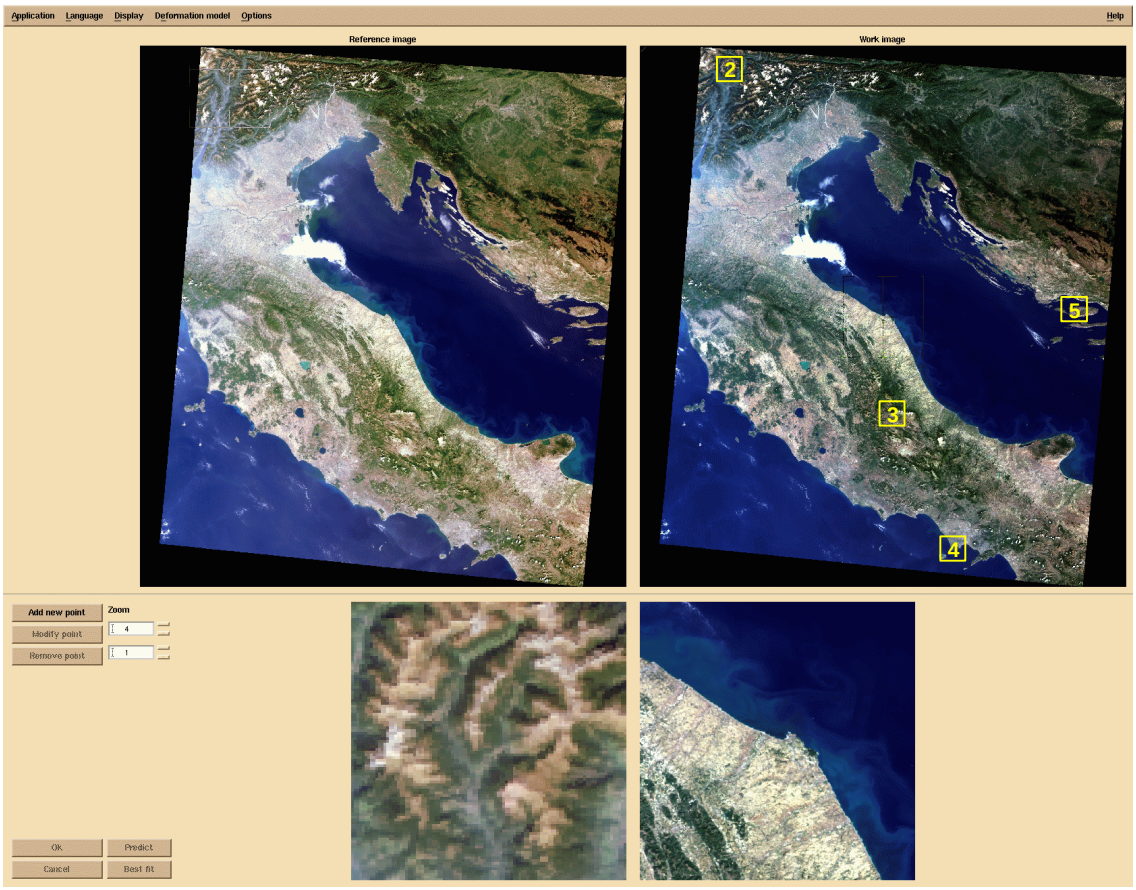


fig. 40 - REGIST application (snap11_REGIST.gif)

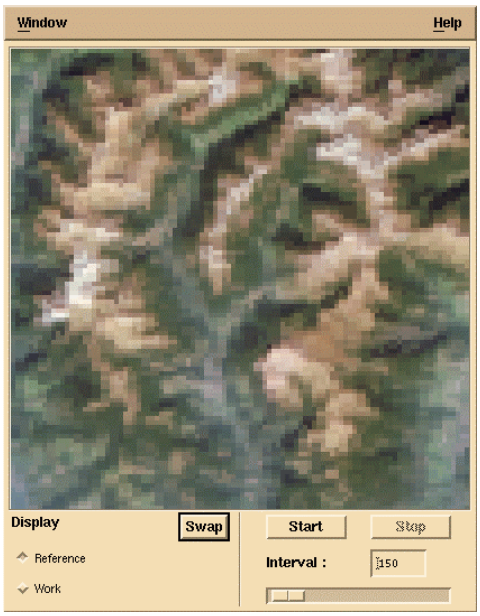


fig. 41 - BEAM (snap12_REGIST_Flicker_x4_beams.gif)

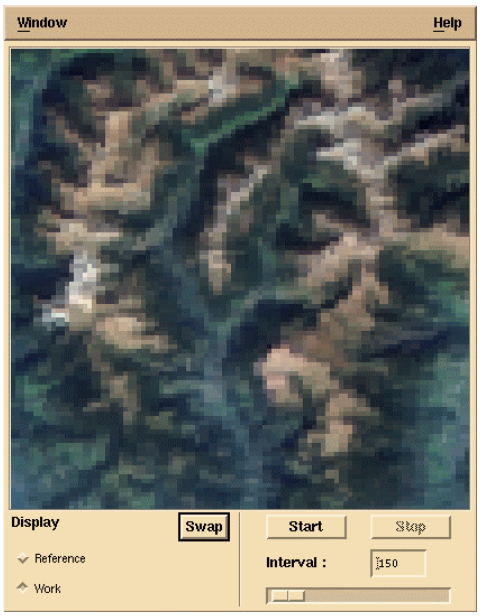


fig. 42 - MERSYN (snap12_REGIST_Flicker_x4_mersyns.gif)

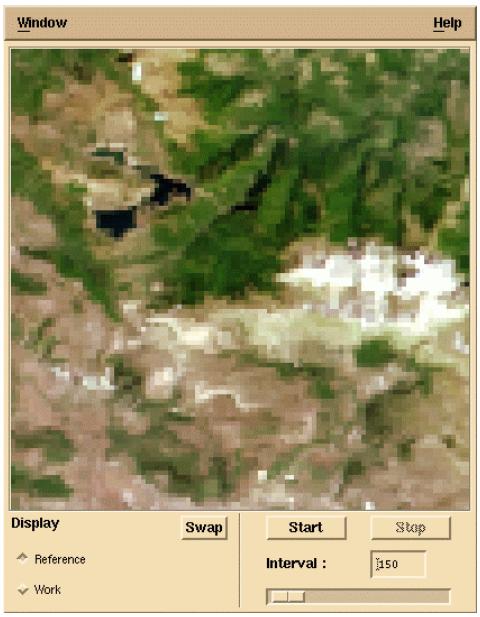


fig. 43 - BEAM (snap13_REGIST_Flicker_x4_beams.gif)

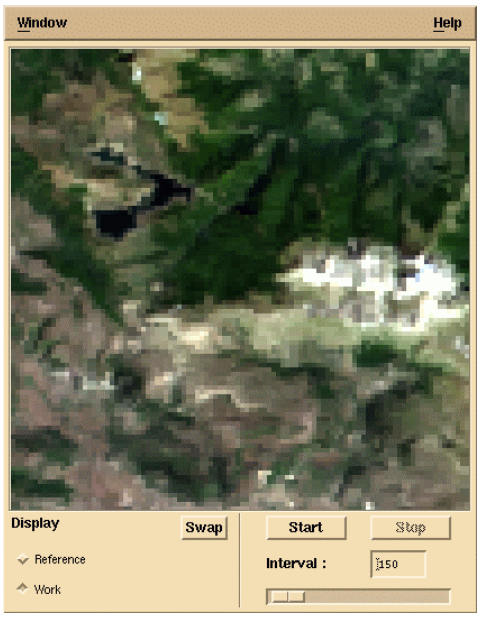


fig. 44 - MERSYN (snap13_REGIST_Flicker_x4_mersyns.gif)

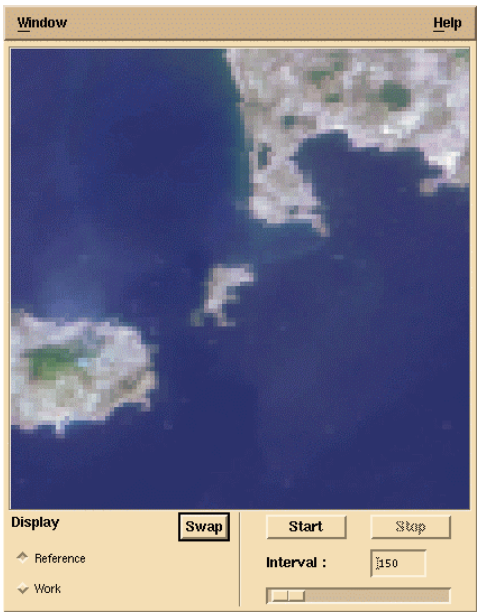


fig. 45 - BEAM (snap04_REGIST_Flicker_14_beams.gif)

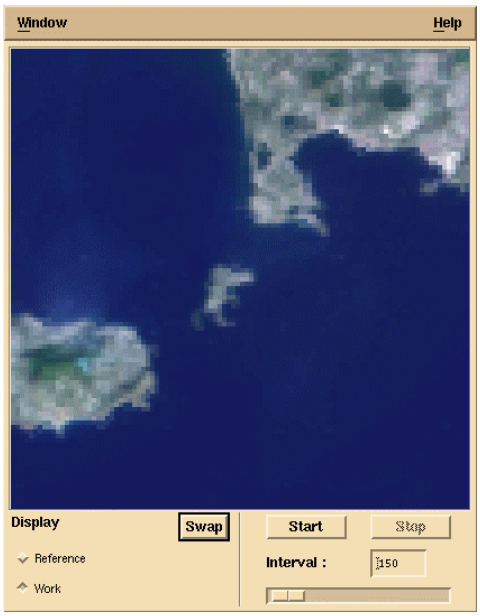


fig. 46 - MERSYN (snap04_REGIST_Flicker_14_mersyns.gif)

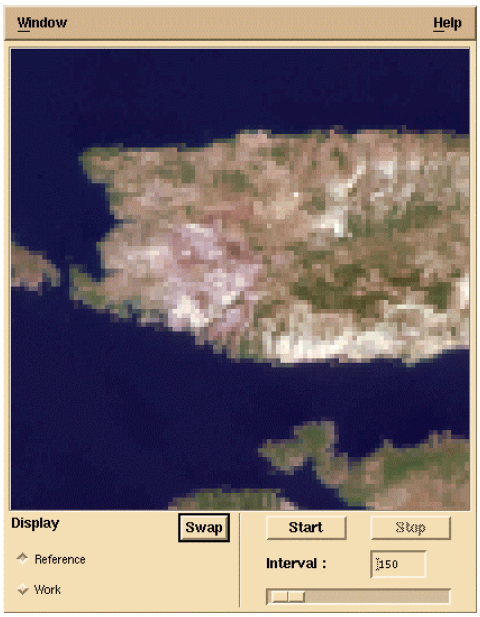


fig. 47 - BEAM (snap05_REGIST_Flicker_14_beams.gif)

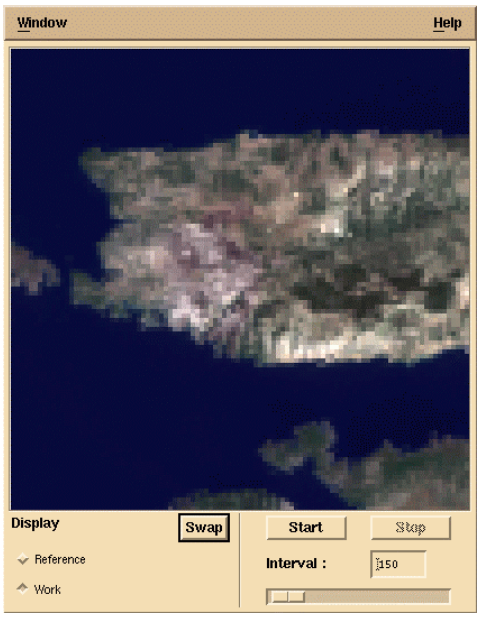


fig. 48 - MERSYN (snap05_REGIST_Flicker_14_mersyns.gif)

4.8 Benchmark

A benchmark of 25 points is provided in APPENDIX D. Points are dispatched within the MERIS synthesis image orthorectified from the GETASSE30 DEM.

5 GETASSE30 QUALITY CONTROL

GETASSE30 has been produced by Marc BOUVET (ESA/ESTEC) after an analysis of the ACE data (see R-5).

"GETASSE30 is a composite dataset. It is using the SRTM30 dataset, ACE dataset, Mean Sea Surface (MSS) data and the EGM96 ellipsoid as sources. The resulting GETASSE30 dataset represents the Earth Topography And Sea Surface Elevation with respect to the WGS84 ellipsoid at 30 arc second resolution." (R-6).

Figure here below (fig. 49 extracted from R-6) shows the flags associated to the origin of GETASSE30 data.

fig. 49 - Origin of GETASSE30 data.

From an e-mail of Steven DELWART dated on 21/10/2004, GETASSE30 data have been extracted from the ESTEC FTP site ftp.estec.esa.int/pub/wipsftp/steven/getasse30/.

GETASS30 data are organised in the following way:

- Tiles of 15°x15° with a resolution of 30'' arc leading to images of 1800x1800 pixels.
- Tile name **nnVeeeH** is based on the coordinates (nn,eee) of the lower-left corner where:
  - nn is the latitude expressed in degrees within [-90°,+90°]
  - V is the hemisphere: N for North or S for South
  - nn is the longitude expressed in degrees within [-180°,+180°]
  - H is the West demi-shere (W) or East demi-sphere (E).
- Elevation values are given in metres above the WGS84 ellipsoid and are stored in 4-bytes ANSI-IEEE floating values.
- Elevations are arranged lines per lines from South to North, and along each line from West to Esat.
- For each tile, two files are given:
  - nnVeeeH containing the elevation data (1800x1800x4 bytes=12960000),
  - nnVeeeH_flag giving the origin of the data (1800x1800x2 unsigned bytes=6480000).

To cover Europe, the following six tiles have been downloaded:

```
-rw-r--r--    1 telimago telimago 12960000 Oct 21 14:06 30N000E.GETASSE30
-rw-r--r--    1 telimago telimago  6480000 Oct 21 13:40 30N000E.GETASSE30_flag
-rw-r--r--    1 telimago telimago 12960000 Nov  1 11:11 30N015E.GETASSE30
-rw-r--r--    1 telimago telimago  6480000 Nov  1 11:11 30N015E.GETASSE30_flag
-rw-r--r--    1 telimago telimago 12960000 Oct 21 13:46 30N015W.GETASSE30
-rw-r--r--    1 telimago telimago  6480000 Oct 21 13:49 30N015W.GETASSE30_flag
-rw-r--r--    1 telimago telimago 12960000 Oct 20 09:42 45N000E.GETASSE30
-rw-r--r--    1 telimago telimago  6480000 Oct 20 09:42 45N000E.GETASSE30_flag
-rw-r--r--    1 telimago telimago 12960000 Nov  1 11:11 45N015E.GETASSE30
-rw-r--r--    1 telimago telimago  6480000 Nov  1 11:11 45N015E.GETASSE30_flag
-rw-r--r--    1 telimago telimago 12960000 Oct 21 13:56 45N015W.GETASSE30
-rw-r--r--    1 telimago telimago  6480000 Oct 21 13:59 45N015W.GETASSE30_flag
```

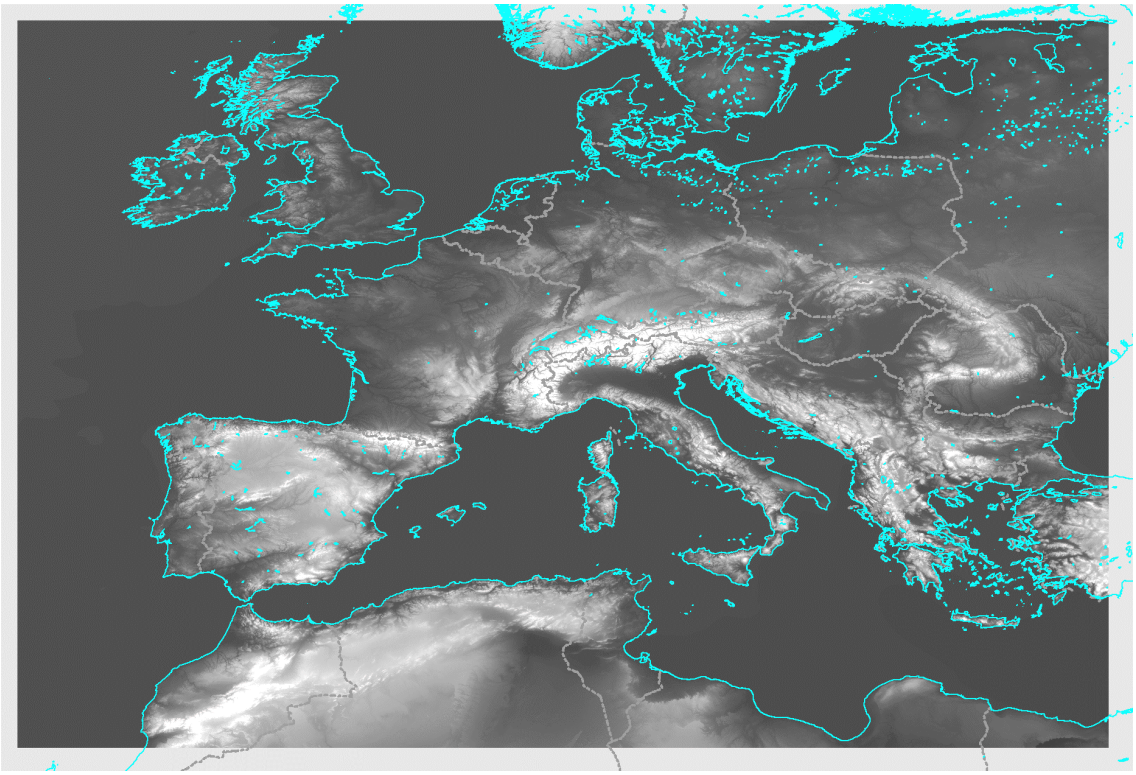


fig. 50 - GETASSE30 over Europe.

In the next sections, GETASSE30 data are compared with:

- EUROPE09, a DEM with a resolution of 9'' arc, and
- a collection of 216 GPS points.

5.1 Comparison with EUROPE09

The 9-seconds Digital Elevation Model of Europe has been one of the most precise and the most used by various organizations of Europe. It is also known as the 250 metres DEM of Europe.

Using the mean Earth radius, the spatial resolution linking the angular resolution and the vertical geodetic distances (or the horizontal ones at equator) is given by the following formula:

$$9'' \text{ arc} \times 6\,370\,997 \text{ m} \approx \underline{278 \text{ m}} \qquad \text{Eq. 6}$$

EUROPË09 has been resampled to achieve the 30'' arc resolution of GETASSE30 and using the WGS84 planimetric reference system (ellipsoid and datum).

Unfortunately, few annotation data are given with EUROPE09 and, in particular, the vertical reference system (WG84, EGM96 or an other one) is unknown. One of the objectives of this section is to determine the closest vertical reference system of EUROPE09 from the one of GETASSE30.

Visual inspection

Shading effect (see fig. 51 and fig. 52) is computed from the first derivative (gain 70 and offset 32768) simulating two illuminations:

- illumination North–West to South-East with a factor 2, and
- illumination North to South with a factor 1.

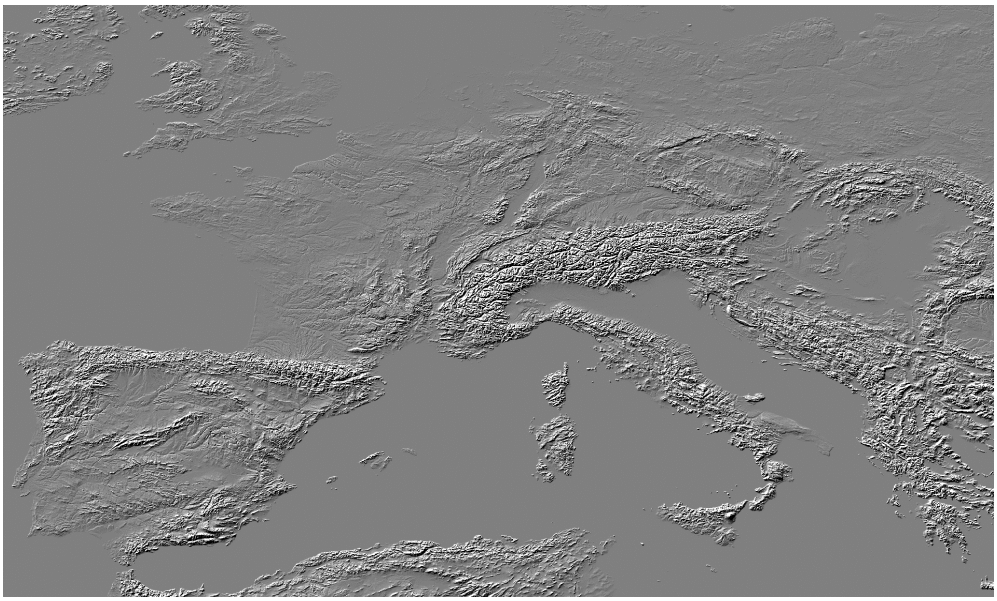


fig. 51 - Shading of GETASSE30 – Full image.

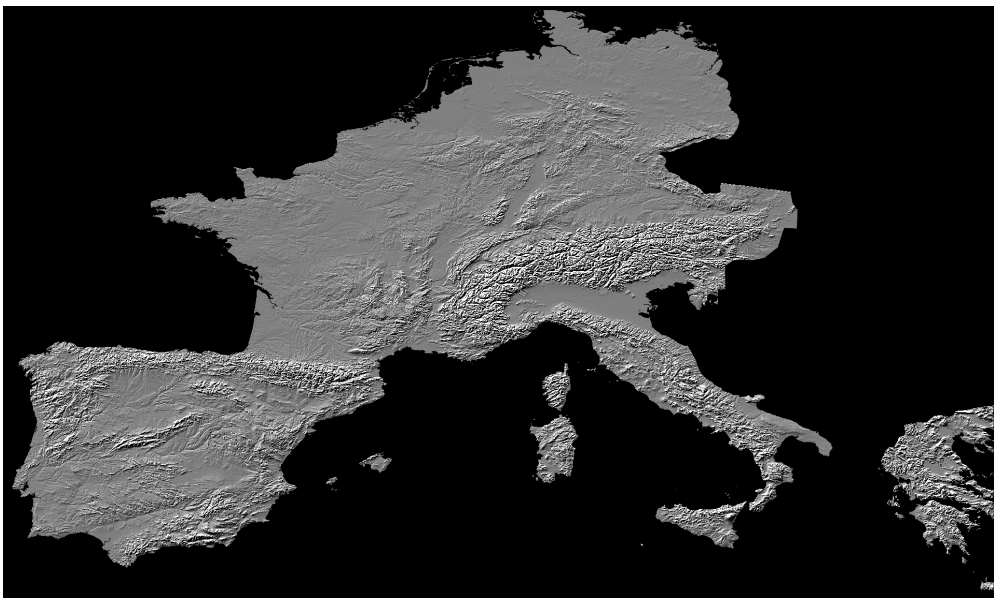


fig. 52 - Shading of EUROPE09 – Full image.

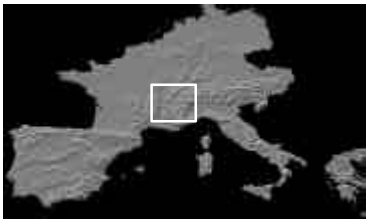
Sea areas of EUROPE09 are set to 0 and are considered as background (represented in black in fig. 52).

At small scale, the morphological structures look exactly the same in both DEMs. On the other hand a full resolution inspection enables identifying small differences when images are alternately displayed (see the animation process as explained in section 4.4).

The two animations “snap32_anim01_switzerland.gif” and “snap33_anim02_spain.gif” are available on demand.

Some of the collected differences are given here below:

1. Two small structures are present only in EUROPE09 (see yellow area of fig. 53).
2. EUROPE09 is moved southward of about 1 pixel (see orange arrow of fig. 53).
3. Roughness of EUROPE09 is higher than GETASSE30 (see in particular in green area of fig. 53).



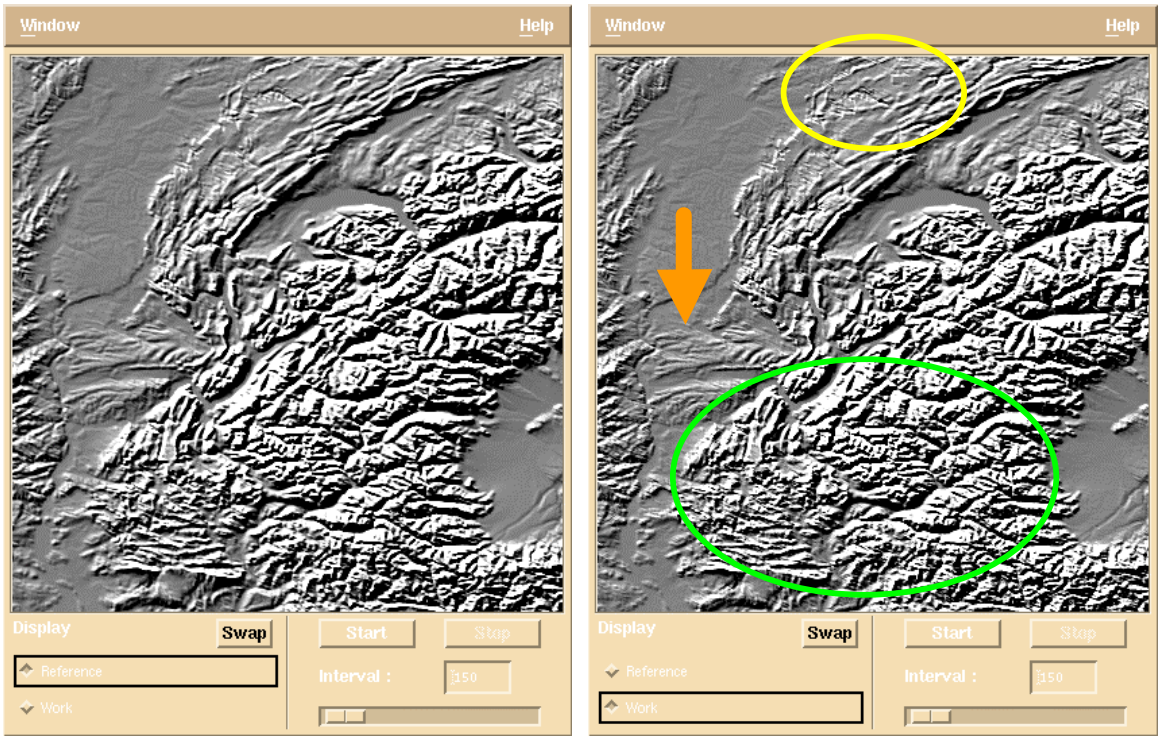


fig. 53 - Switzerland animation 01. GETASSE30 (left) and EUROPE09 (right).

On an other region of South of Spain (see fig. 54) processing an animation underlines the following defects:

4. Stitches between probable tiles within EUROPE09 (see blue area of fig. 54).

5. EUROPE09 is moved southward of about 1 pixel (see orange arrow of fig. 54).
6. Roughness of EUROPE09 is higher than GETASSE30.

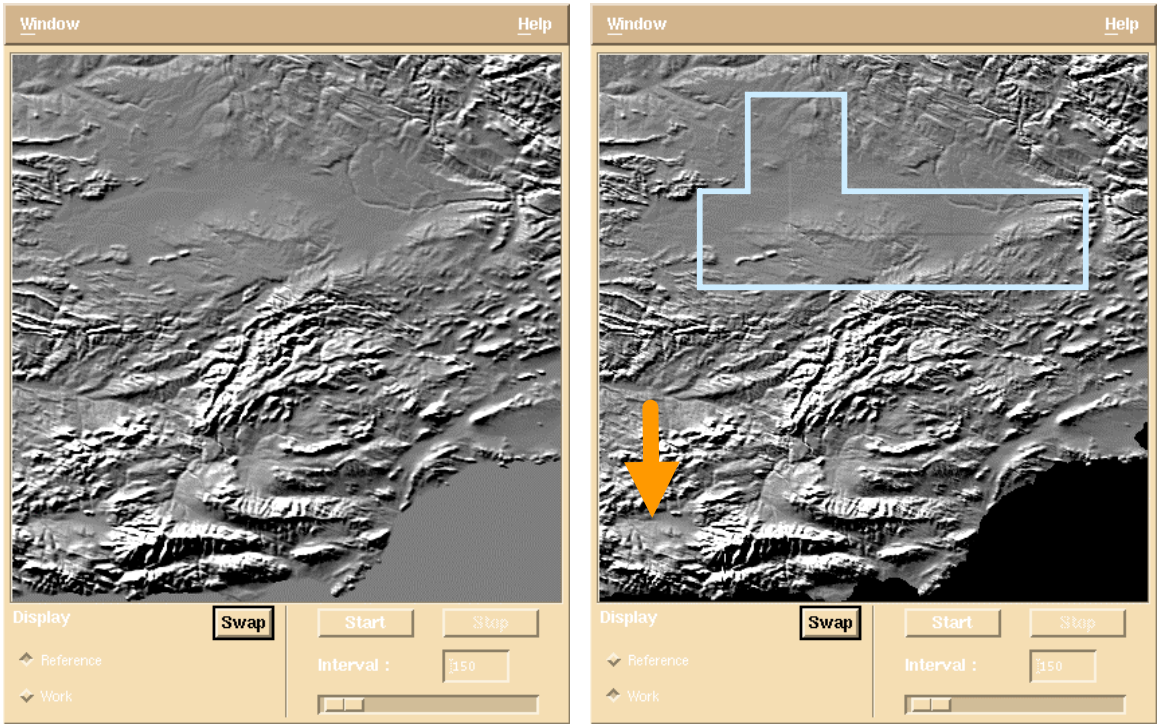


fig. 54 - Spain animation 02. GETASSE30 (left) and EUROPE09 (right).

Statistics of image difference

GETASSE30 elevations are given above the WGS84 ellipsoid (R-6).

In order to determine the most probable vertical reference system in which EUROPE09 elevations are given, GETASSE30 has been also computed above EGM96 geoide, subtracting the EGM96 image downloaded from:

http://cddisa.gsfc.nasa.gov/926/egm96/egm96.html
-> http://earth-info.nga.mil/GandG/wgsegm/
-> http://earth-info.nima.mil/GandG/wgsegm/egm96.html

.

This section shows result images and statistics when computing the differences:

1. GETOPO30_EGM96 – EUROPE09, and
2. GETOPO30_WGS84 – EUROPE09.

GETOPO30_EGM96 – EUROPE09

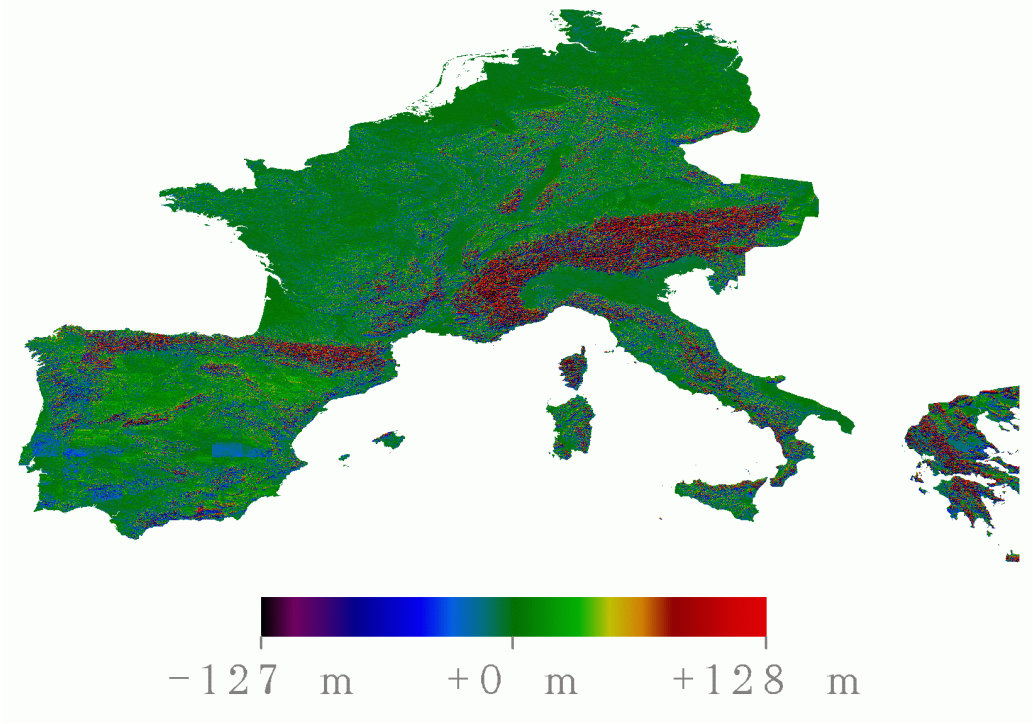
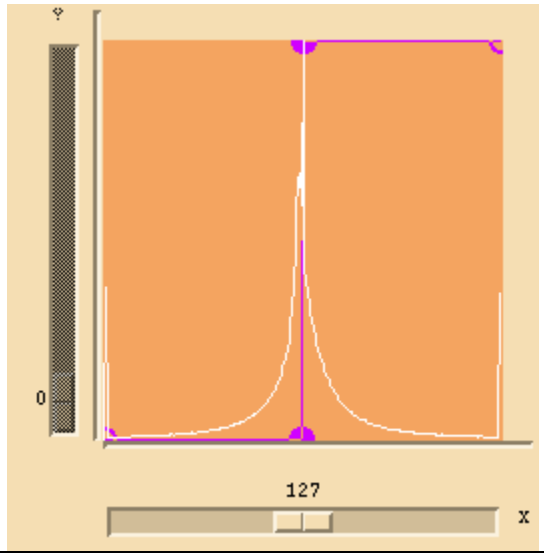


fig. 55 - EUROPE09 subtracted to GETASSE30 above EGM96 – Image of differences.



ORIGINAL VALUES:

GAIN TO BE APPLIED	=	1.000000
OFFSET TO BE APPLIED	=	-32768.000000
BACKGROUND VALUE	=	-32768.00000
MINIMUM	=	-1152.000
MAXIMUM	=	1267.000
LEFT BOUND 2%	=	-129.000
RIGHT BOUND 2%	=	128.000
MEAN	**=**	**0.429**
VARIANCE	=	2773.290
STANDARD DEVIATION	**=**	**52.662**
RMS	=	52.664

fig. 56 - EUROPE09 subtracted to GETASSE30 above EGM96 – Statistics.

GETOPO30_WGS84 – EUROPE09

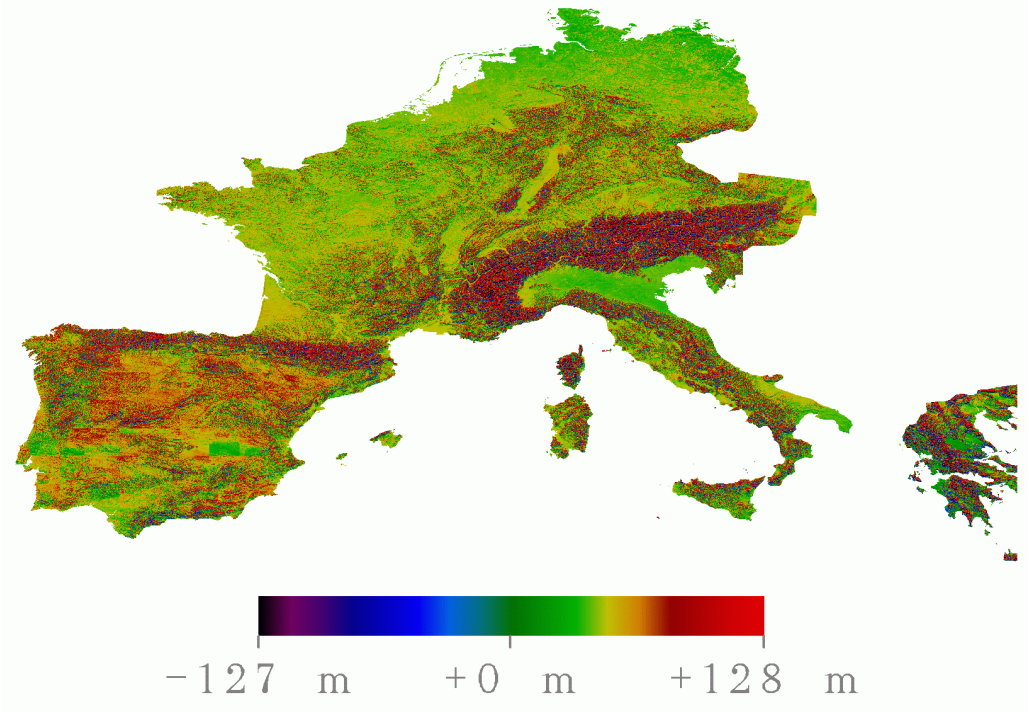
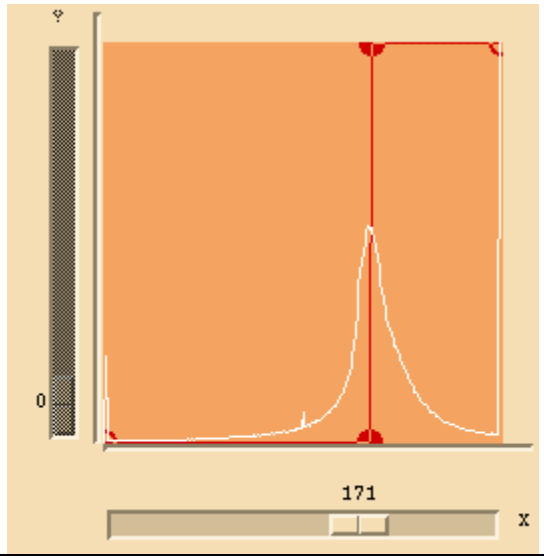


fig. 57 - EUROPE09 subtracted to GETASSE30 above WGS84 – Image of differences.




```
ORIGINAL VALUES:

   GAIN TO BE APPLIED      = 1.000000
   OFFSET TO BE APPLIED    = -32768.000000

   BACKGROUND VALUE        = -32768.00000

   MINIMUM                 =   -1102.000
   MAXIMUM                 =    1317.000

   LEFT  BOUND 2%          =     -82.000
   RIGHT BOUND 2%          =     177.000

   MEAN                    =      48.723
   VARIANCE                =    2813.261
   STANDARD DEVIATION      =      53.040

   RMS                     =      72.022
```

fig. 58 - EUROPE09 subtracted to GETASSE30 above WGS84 – Statistics.

EUROPE09 vertical reference system

EUROPE09 is much closer to GETASSE30 above EGM96 (difference mean of 0.429 metres) than to GETASSE30 above WGS84 (difference mean of 48.723 metres). This means that EUROPE09 shall have been expressed above a geoide (for example EGM96) or above national geoids for each country.

Distribution of differences

Distribution of differences seems to obey a normal distribution leading to a Gaussian histogram with a standard deviation of +/-53 metres.

Defect identification

Stretching the difference between the two DEMs magnifies the defect identification. For example, fig. 59 clearly underlines the tile defect observed in the DEM of Spain. More subtle defects may also be accounted like the punctual anomalies illustrated in fig. 60.

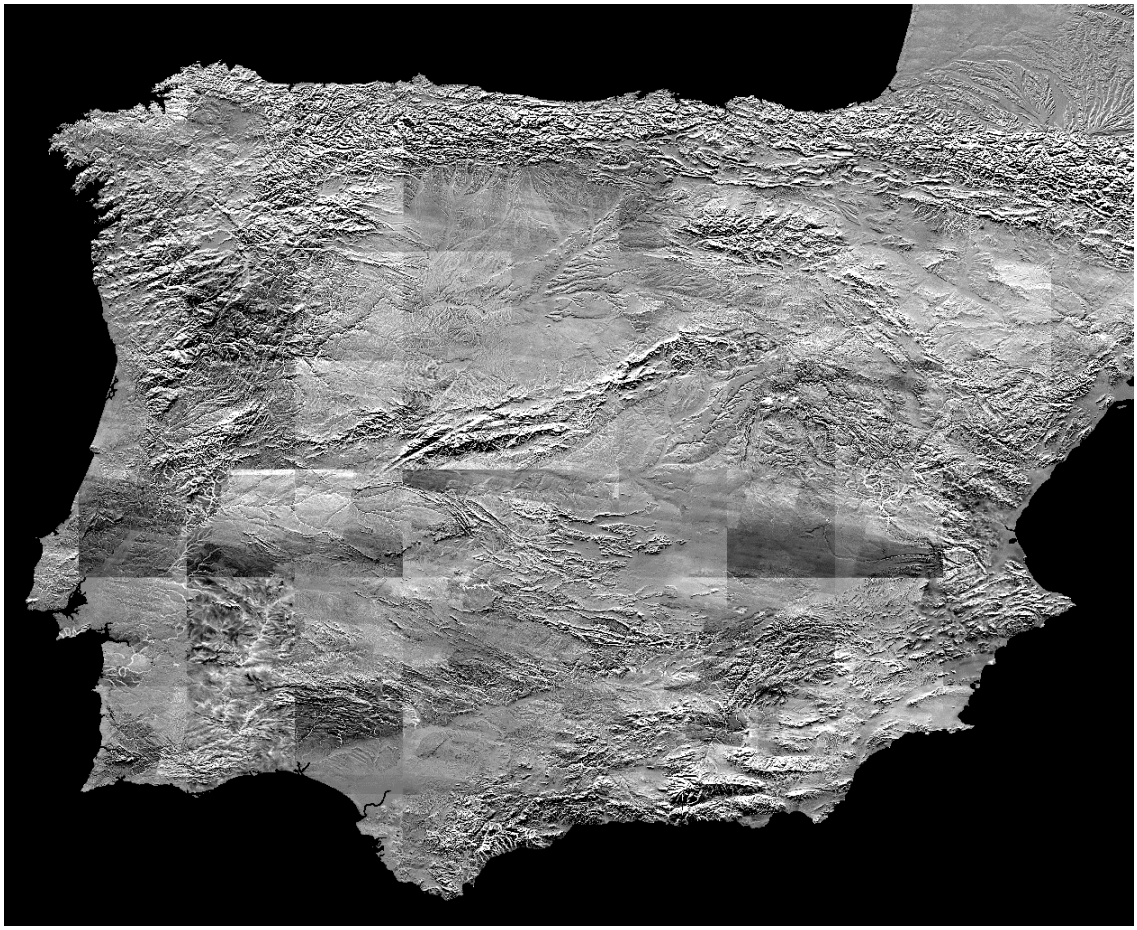


fig. 59 - GETOPO30_EGM96–EUROPE09 – Tile defect.

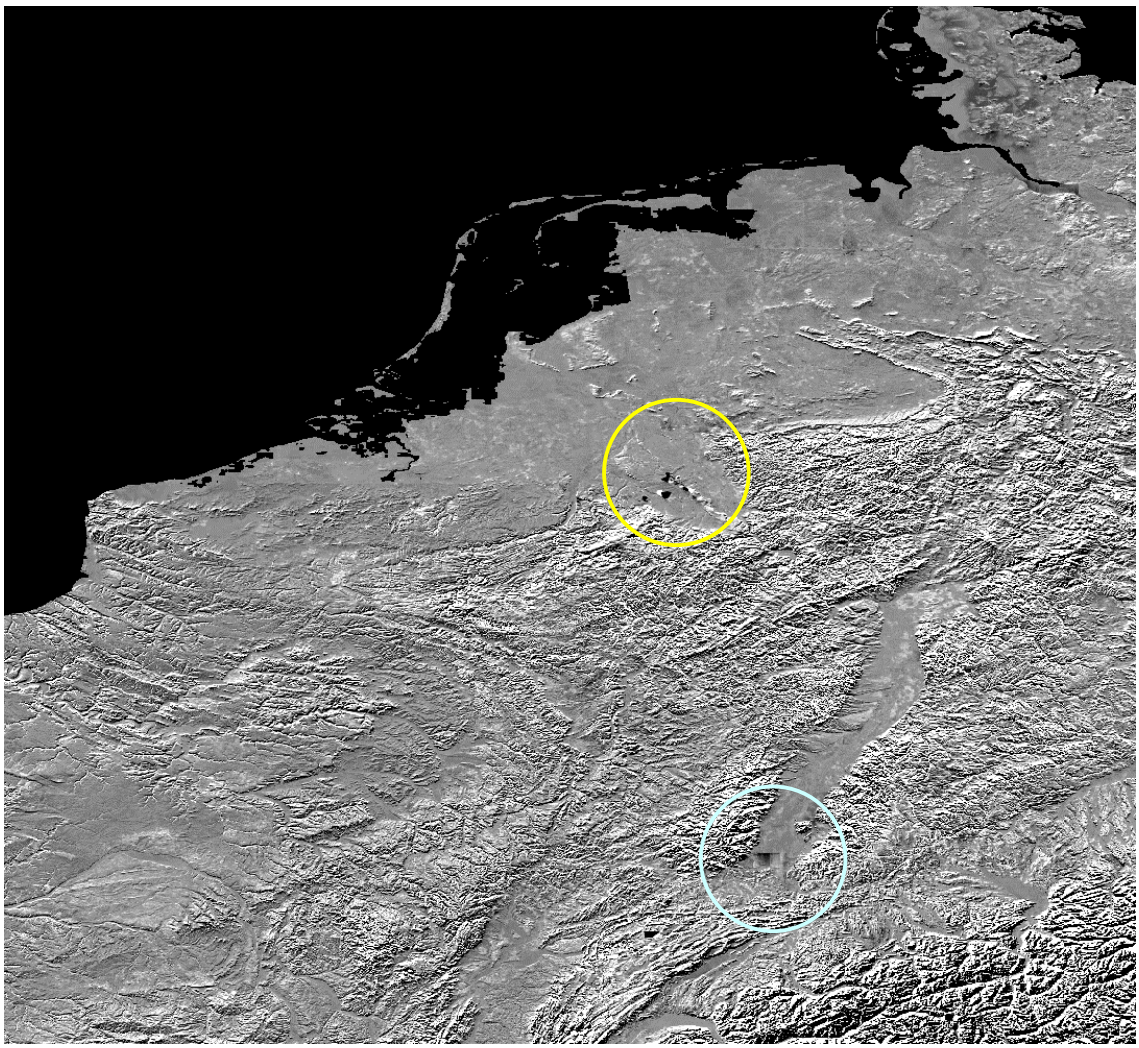


fig. 60 - GETOPO30_EGM96–EUROPE09 – Punctual defects.

5.2 Comparison with EUVN geodesic points

Description of reference data

Geodesic points

Geodesic points are available at the address http://crs.bkg.bund.de/evrs/Results.html of the European Vertical Reference System (EVRS) page shown in figure here below.

This site is maintained by the EUREF (see http://www.euref-iag.net/).

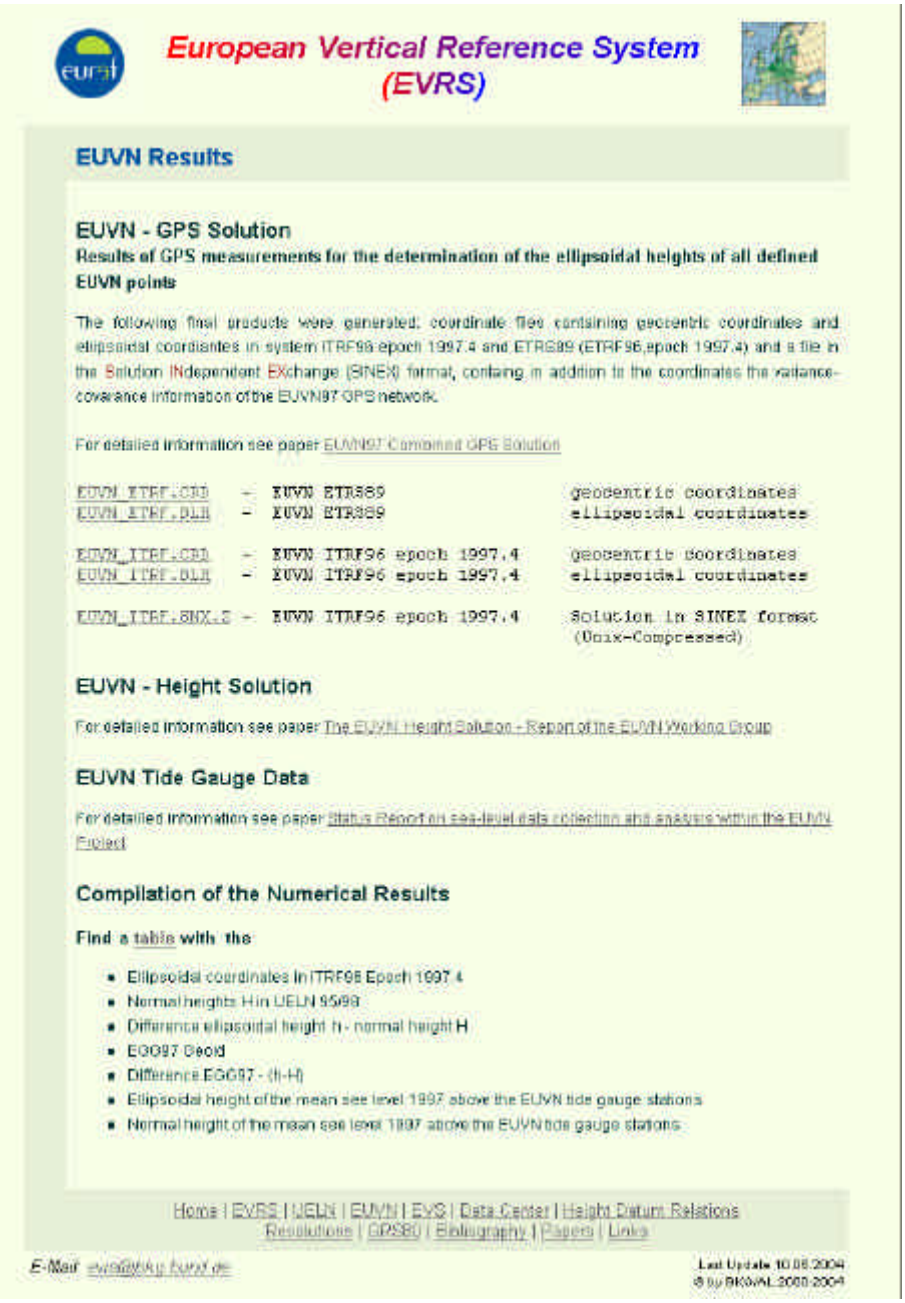


fig. 61 - Site providing a collections of geodesic points.

File EUVN_ETRF.BLH contains 217 points that are distributed over Europe as shown by fig. 62.

fig. 62 - Distribution of GPS points over Europe.

ITRS and WGS84

ITRS (International Terrestrial Reference System) and WGS84 (World Geodetic System 1984) are almost equivalent (see http://www.ensg.ign.fr/ITRF/trans_para.php here below).

```
In general the ITRS (and its realizations ITRFyy) are identical to WGS84
at one meter level.

Meanwhile there are two types of WGS84 realization:

- old realization based on U.S. Navy Navigation Satellite System,
  commonly known as DOPPLER Transit, and provided station coordinates
  with accuracies of about one meter. With respect to this realization
  we published, some years ago, transformation parameters between
  ITRF90 and this Doppler realized system:

Parameters from ITRF90 to WGS84-Doppler realized system

             T1      T2      T3       D         R1       R2        R3
UNITS -----> (m)     (m)     (m)     (ppm)     (")      (")       (")
------------------------------------------------------------------------
            0.060   -0.517  -0.223  -0.011    0.0183  -0.0003    0.0070


- New realizations of WGS84 based on GPS data, such as WGS84(G730 or
G873).
  These new WGS84 realizations are coincident with ITRF at about
  10-centimeter level. For these realizations there are no official
```

```
transformation parameters. This means that one can consider that
ITRF coordinates are also expressed in WGS84 at 10 cm level.
```

A French explanation of ITRS is provided in a document of IGN at URL address http://www.ign.fr/telechargement/ education/fiches/geodesie/systèmes.pdf.

```
L'ITRS, système de référence terrestre de l'International Earth Rotation Service
(IERS), est matérialisé par un réseau mondial de près de 300 points. Avec une
exactitude au niveau centimétrique, il s'agit du plus précis des systèmes
géodésiques mondiaux. Depuis 1988, l'IERS fournit chaque année une réalisation
appelée ITRFyy, avec yy comme derniers chiffre du millésime. La dernière en date
est l'ITRF97.

Du fait de la présicion de ce système, l'époque de référence des coordonnées qui
sont publiées doit être précisée (par exemple : 1997.0 pour l'ITRF97). De plus,
les réalisations ITRFnn donnent pour chaque point une vitesse de déplacement, qui
est une mesure du mouvement de ce point résultant de phénomèmes géophysiques
comme les mouvements tectoniques et le rebond post-glaciaire. D'autre part, on
applique lors du calcul de ces coordonnées des corrections diverses : marées
terrestres, pression atmosphérique, etc...

Les réalisations ITRFnn sont obtenues par combinaison de jeux de coordonnées
issues de différentes techniques de géodésie spatiale très précises :

VLBI : Very Long Base Interferometry (interférométrie à très longue base) LLR :
Lunar Laser Ranging (télémétrie laser sur la lune) SLR : Satellite Laser Ranging
(télémétrie laser sur satellite) GPS : Global positioning System DORIS : Doppler
Orbitography Radiopositionning Integrated by Satellite

Le système ETRS89 (European Terrestrial Reference System 1989)

Le système ETRS89 est défini a partir de l'ITRS et coïncide avec ITRS à l'époque
1989.0.

Il est attaché à la partie stable de la plaque eurasienne. A une époque
quelconque on applique une vitesse théorique de la plaque Eurasie sur les
coordonnées.

L'ETRFyy, réalisation de ETRS utilise des points ITRFyy européens et des points
de densification par GPS. Successeur de ED87, initialement localisé sur l'Europe
occidentale, ETRS89 s'étend à présent sur l'Europe de l'Est.
```

More information can be found on the following sites:

- ITRF http://www.ensg.ign.fr/ITRF/
- Geodesy tutorial http://www.qgsl.com/geodesy_tutorial/intro.php

GTASSE30 Vertical Reference System

As indicated in the "**Readme_with_images.doc**" file of the GETASSE30 distribution, this DEM contains elevation expressed above the WGS84 ellipsoid.

```
…
============================================================
                        GETASSE30

Global Earth Topography and Sea Surface Elevation at 30 arc second resolution

============================================================

Data set:
```

GETASSE30 is a composite dataset. It is using the SRTM30 dataset, ACE dataset, Mean Sea Surface (MSS) data and the EGM96 ellipsoid as sources.
The resulting GETASSE30 dataset represents the Earth Topography And Sea Surface Elevation with respect to the WGS84 ellipsoid at 30 arc second resolution.
…

EUVN ground control points are GPS measurements expressed in reference to the ETRS89 or ITRF96 reference systems that are, as shown in the previous section, very close the WGS84 reference system.

As a consequence, elevations of EUVN geodesic points and those found in GETASSE30 are expressed with respect to very close vertical reference systems, and may be immediately compared without performing a reference system conversion (see for example the HEIGHT process to add/substract geoid elevations above the WGS84 ellipsoid).

Quality control results

The full logs of quality controls are stored in APPENDIX F.

For each one of the 217 geodesic point (X_i,Y_i,H_i), DEMCHK process checks if the point (X_i,Y_i) is inside the DEM or not. In the positive case, the corresponding elevation h_i is interpolated from the DEM to be controlled.

Statistics are computed according to the following formulae:

$$\textit{ARITHMETIC MEAN} = \frac{1}{N}\times\sum_{i=1}^{N}(H_i - h_i)$$

$$\textit{QUADRATIC MEAN} = \sqrt{\frac{1}{N}\times\sum_{i=1}^{N}(h_i - H_i)^2}$$

$$\textit{STANDARD DEVIATION} = \sqrt{\frac{1}{N}\times\sum_{i=1}^{N}(h_i - H_i)^2 - \left[\frac{1}{N}\times\sum_{i=1}^{N}(h_i - H_i)\right]^2}$$

$$\textit{MINIMUM} = \min_{i=1..N}(H_i - h_i)$$

$$\textit{MAXIMUM} = \max_{i=1..N}(H_i - h_i)$$

Interpolated value h_i depends on the interpolation method being used. The three classical methods (*Nearest neighbour*, *Bi-linear* and *Bi-cubic*) have been tested.

Checking has been applied to the GTASSE30 DEM over Europe (see fig. 50). On this area **169 EUVN control points** among the 217 available ones fail inside the DEM (77.9% of the points).

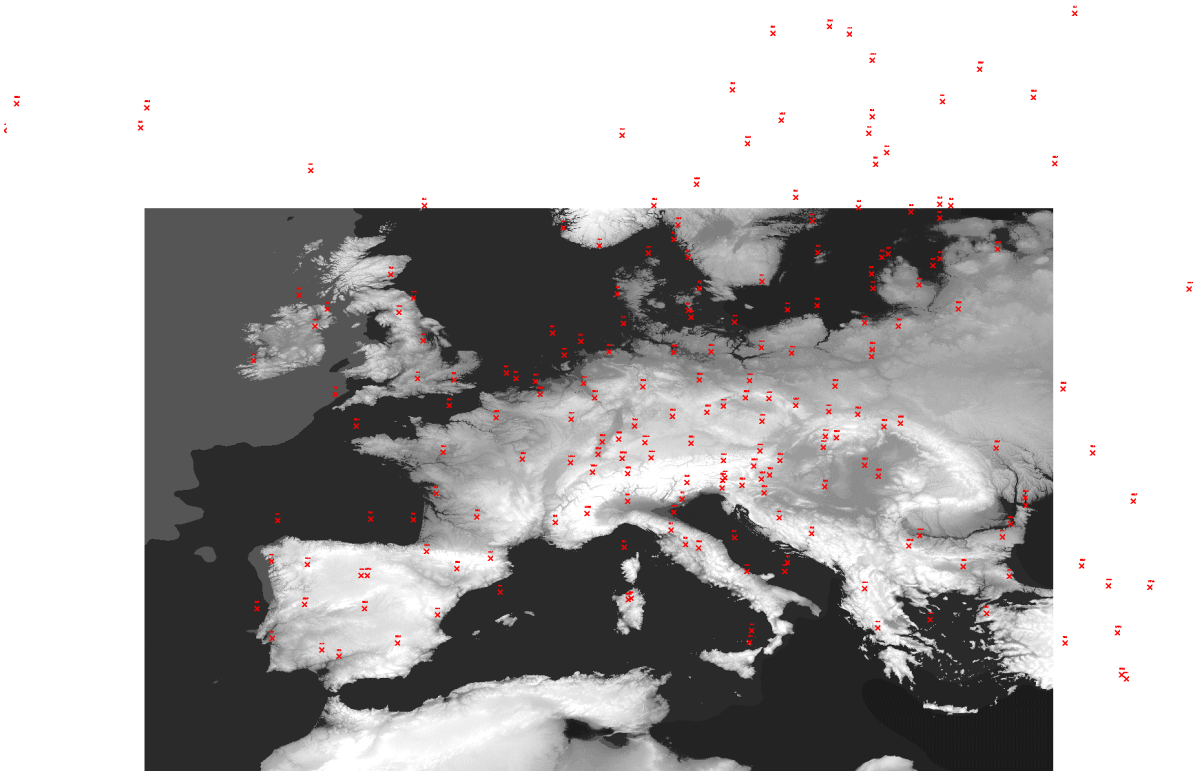


fig. 63 - Distribution of the geodesic control points relatively to GETASSE30 DEM over Europe.

Table here below gives results of the controls performed on GTASSE30 Europe DEM for the three interpolation methods (“Nearest neighbour”, “Bi-linear”, and “Bi-cubic”) and a control performed on ACE DEM using the “Nearest neighbour” interpolation method.

	GTASSE30 Nearest neighbour	GTASSE30 Bi-linear	GTASSE30 Bi-cubic	ACE Nearest neighbour
ARITHMETIC MEAN	+10.828	+11.304	+12.008	-2201.149
QUADRATIC MEAN	**32.368**	**34.017**	**32.915**	**12109.911**
STANDARD DEVIATION	30.503	32.083	30.646	11908.186
MINIMUM	-127.560	-181.091	-172.570	-65001.139
MAXIMUM	+100.931	+125.763	127.387	+325.560

table 4 - Statistics of elevation differences (expressed in metres).

Surprinsingly, the “Nearest neughbour” interpolation leads to the lowest error estimation. Such an observation may be due to the error introduced by whatever interpolation when considering the surrounding pixels. These classical interpolation methods are not relevant to topographic surfaces that may be complex.

Controls performed on ACE lead to signifi errors. These statistical results for ACE are not significant because of specific values found within the ACE DEM file and that are not elevations (see appendix F.1). A full analysis of ACE would require to check the ACE image of flags indicating if the elevation is relevant or not and the origin of this elevation.

Figures below show the distribution of elevation errors for the various interpolation methods. We may note that the distribution is almost symmetrical towards a vertical axis located at abscise almost given by the arithmetic mean. (≈+11).

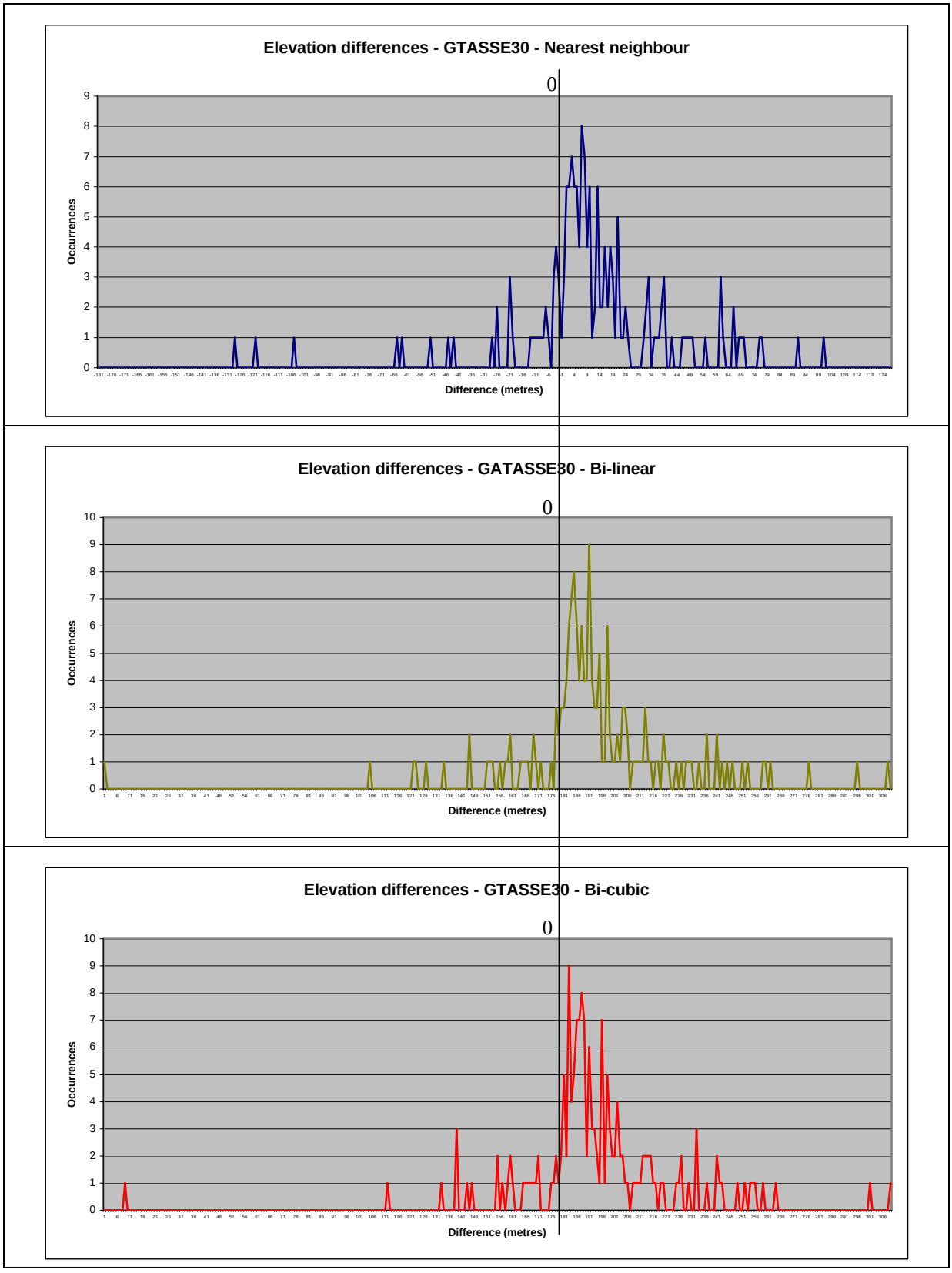


fig. 64 - Elevation errors distribution for various interpolation methods.

APPENDIX A MERIS SCENE FR ITALY

A.1 Decoding log

```
MAIN PRODUCT HEADER:
Product   File   Name                                                                :
MER_FR__1PNUPA20030921_092341_000000982020_00079_08149_0534.N1
Processing Stage Flag                     : N
Reference Document Describing Product     : PO-RS-MDA-GS2009_11_3J
Acquisition Station ID                    : PDHS-E
Processing Center ID                      : UK-PAC
UTC Time of Processing                    : 26/01/2004  13:24:19.434677
Software Version Number of Processing SW  : MERIS/4.07
UTC Start Time of Data Sensing            : 21/09/2003  09:22:17.265770
UTC Stop Time of Data Sensing             : 21/09/2003  09:26:36.187634
Phase                                     : 2
Cycle                                     : 20
Start Relative Orbit Number               : 79
Start Absolute Orbit Number               : 8149
State vector at ascending node:
   Date Time                              : 21/09/2003  09:23:00.000000
   Position                               : 4513681.740000  1552817.947000  5333228.142000
   Velocity                               : 5786.205174  -136.373816  -4845.448794
DUT1=UT1-UTC                              : -0.352582
Source of Orbit Vectors                   : FR (FOS restituted)
UTC time corresponding to SBT below       : 21/09/2003  09:10:16.414244
Satellite Binary Time (SBT)               : 4008373760
Clock Step Size (in picoseconds)          : 0.003906249810
UTC time of occurrence of Leap Second     : 17/10/2001  00:00:00.000000
Leap second sign                          : 1
Leap second error                         : 0
Product error                             : 0
Total Size of Product                     : 166271715
Length of SPH                             : 9942
Number of DSDs                            : 30
Length of each DSD                        : 280
Number of DSs attached                    : 19


SPECIFIC PRODUCT HEADER:
SPH Descriptor                            : MER_FR__1P SPECIFIC HEADER
Stripline Continuity Indicator            : 0
Slice Position                            : 1
Number of Slices in this Stripline        : 1
Azimuth time first line of product        : 21/09/2003  09:23:41.714409
Azimuth time last line of product         : 21/09/2003  09:25:20.267689
Geo. Lat. of first sample of first line   : 45.90986300
Geo. Long. of first sample of first line  : 18.01469200
Geo. Lat. of middle sample of first line  : 46.52906400
Geo. Long. of middle sample of first line : 14.34706300
Geo. Lat. of last sample of first line    : 47.02819600
Geo. Long. of last sample of first line   : 10.60349600
Geo. Lat. of first sample of last line    : 40.14283200
Geo. Long. of first sample of last line   : 15.96981400
Geo. Lat. of middle sample of last line   : 40.73596700
Geo. Long. of middle sample of last line  : 12.62631800
Geo. Lat. of last sample of last line     : 41.23065400
Geo. Long. of last sample of last line    : 9.22799100
Transmission error flag                   : 0
Format error flag                         : 0
Database error flag                       : 0
Coarse error flag                         : 0
Forecast (0) / Analysis (1) ECMWF data    : 1
Number of transmission errors in product  : 0
Number of format errors in the product    : 0
Threshold for setting TRANS_ERR_FLAG      : 0.000000 %
Threshold for setting FORMAT_ERR_FLAG     : 0.000000 %
```

```
Number of bands in the product          : 15
Central wavelength of the bands:
   Band 1                               : 412.545 nm
   Band 2                               : 442.401 nm
   Band 3                               : 489.744 nm
   Band 4                               : 509.700 nm
   Band 5                               : 559.634 nm
   Band 6                               : 619.620 nm
   Band 7                               : 664.640 nm
   Band 8                               : 680.902 nm
   Band 9                               : 708.426 nm
   Band 10                              : 753.472 nm
   Band 11                              : 761.606 nm
   Band 12                              : 778.498 nm
   Band 13                              : 864.833 nm
   Band 14                              : 884.849 nm
   Band 15                              : 899.860 nm
Bandwidth of the bands:
   Band 1                               : 9.930 nm
   Band 2                               : 9.946 nm
   Band 3                               : 9.965 nm
   Band 4                               : 9.971 nm
   Band 5                               : 9.983 nm
   Band 6                               : 9.991 nm
   Band 7                               : 9.994 nm
   Band 8                               : 7.493 nm
   Band 9                               : 9.996 nm
   Band 10                              : 7.493 nm
   Band 11                              : 3.742 nm
   Band 12                              : 15.000 nm
   Band 13                              : 19.999 nm
   Band 14                              : 9.990 nm
   Band 15                              : 9.989 nm
Instantaneous Field of View             : 0.019159 deg.
Processor mode of oper. (1:Raw,0:Full)  : 1
Offset compensation flag (1:Yes,0:No)   : 1
Line spacing in time                    : 0.043997 s.
Number of samples per output line       : 2241
Nb. of lines between along track tie pts : 64
Nb. samples between across track tie pts : 64
On ground spacing between columns       : 260.000 m.


DATA SET DESCRIPTOR #1:
Data Set Name                           : Quality ADS
Data Set Type                           : A
External Product Reference              :
DS Offset in bytes                      : 11189
Total Size of DS in bytes               : 165
Number of DSRs within the DS            : 5
Length of the DSRs in bytes             : 33


DATA SET DESCRIPTOR #2:
Data Set Name                           : Scaling Factor GADS
Data Set Type                           : G
External Product Reference              :
DS Offset in bytes                      : 11354
Total Size of DS in bytes               : 292
Number of DSRs within the DS            : 1
Length of the DSRs in bytes             : 292


DATA SET DESCRIPTOR #3:
Data Set Name                           : Tie points ADS
Data Set Type                           : A
External Product Reference              :
DS Offset in bytes                      : 11646
Total Size of DS in bytes               : 65268
Number of DSRs within the DS            : 36
Length of the DSRs in bytes             : 1813
```

```
DATA SET DESCRIPTOR #4:
Data Set Name                          : Radiance MDS(1)
Data Set Type                          : M
External Product Reference             :
DS Offset in bytes                     : 76914
Total Size of DS in bytes              : 10073295
Number of DSRs within the DS           : 2241
Length of the DSRs in bytes            : 4495


DATA SET DESCRIPTOR #5:
Data Set Name                          : Radiance MDS(2)
Data Set Type                          : M
External Product Reference             :
DS Offset in bytes                     : 10150209
Total Size of DS in bytes              : 10073295
Number of DSRs within the DS           : 2241
Length of the DSRs in bytes            : 4495


DATA SET DESCRIPTOR #6:
Data Set Name                          : Radiance MDS(3)
Data Set Type                          : M
External Product Reference             :
DS Offset in bytes                     : 20223504
Total Size of DS in bytes              : 10073295
Number of DSRs within the DS           : 2241
Length of the DSRs in bytes            : 4495


DATA SET DESCRIPTOR #7:
Data Set Name                          : Radiance MDS(4)
Data Set Type                          : M
External Product Reference             :
DS Offset in bytes                     : 30296799
Total Size of DS in bytes              : 10073295
Number of DSRs within the DS           : 2241
Length of the DSRs in bytes            : 4495


DATA SET DESCRIPTOR #8:
Data Set Name                          : Radiance MDS(5)
Data Set Type                          : M
External Product Reference             :
DS Offset in bytes                     : 40370094
Total Size of DS in bytes              : 10073295
Number of DSRs within the DS           : 2241
Length of the DSRs in bytes            : 4495


DATA SET DESCRIPTOR #9:
Data Set Name                          : Radiance MDS(6)
Data Set Type                          : M
External Product Reference             :
DS Offset in bytes                     : 50443389
Total Size of DS in bytes              : 10073295
Number of DSRs within the DS           : 2241
Length of the DSRs in bytes            : 4495


DATA SET DESCRIPTOR #10:
Data Set Name                          : Radiance MDS(7)
Data Set Type                          : M
External Product Reference             :
DS Offset in bytes                     : 60516684
Total Size of DS in bytes              : 10073295
Number of DSRs within the DS           : 2241
Length of the DSRs in bytes            : 4495


DATA SET DESCRIPTOR #11:
Data Set Name                          : Radiance MDS(8)
Data Set Type                          : M
```

```
External Product Reference              :
DS Offset in bytes                      : 70589979
Total Size of DS in bytes               : 10073295
Number of DSRs within the DS            : 2241
Length of the DSRs in bytes             : 4495


DATA SET DESCRIPTOR #12:
Data Set Name                           : Radiance MDS(9)
Data Set Type                           : M
External Product Reference              :
DS Offset in bytes                      : 80663274
Total Size of DS in bytes               : 10073295
Number of DSRs within the DS            : 2241
Length of the DSRs in bytes             : 4495


DATA SET DESCRIPTOR #13:
Data Set Name                           : Radiance MDS(10)
Data Set Type                           : M
External Product Reference              :
DS Offset in bytes                      : 90736569
Total Size of DS in bytes               : 10073295
Number of DSRs within the DS            : 2241
Length of the DSRs in bytes             : 4495


DATA SET DESCRIPTOR #14:
Data Set Name                           : Radiance MDS(11)
Data Set Type                           : M
External Product Reference              :
DS Offset in bytes                      : 100809864
Total Size of DS in bytes               : 10073295
Number of DSRs within the DS            : 2241
Length of the DSRs in bytes             : 4495


DATA SET DESCRIPTOR #15:
Data Set Name                           : Radiance MDS(12)
Data Set Type                           : M
External Product Reference              :
DS Offset in bytes                      : 110883159
Total Size of DS in bytes               : 10073295
Number of DSRs within the DS            : 2241
Length of the DSRs in bytes             : 4495


DATA SET DESCRIPTOR #16:
Data Set Name                           : Radiance MDS(13)
Data Set Type                           : M
External Product Reference              :
DS Offset in bytes                      : 120956454
Total Size of DS in bytes               : 10073295
Number of DSRs within the DS            : 2241
Length of the DSRs in bytes             : 4495


DATA SET DESCRIPTOR #17:
Data Set Name                           : Radiance MDS(14)
Data Set Type                           : M
External Product Reference              :
DS Offset in bytes                      : 131029749
Total Size of DS in bytes               : 10073295
Number of DSRs within the DS            : 2241
Length of the DSRs in bytes             : 4495


DATA SET DESCRIPTOR #18:
Data Set Name                           : Radiance MDS(15)
Data Set Type                           : M
External Product Reference              :
DS Offset in bytes                      : 141103044
Total Size of DS in bytes               : 10073295
```

```
Number of DSRs within the DS            : 2241
Length of the DSRs in bytes             : 4495


DATA SET DESCRIPTOR #19:
Data Set Name                           : Flags MDS(16)
Data Set Type                           : M
External Product Reference              :
DS Offset in bytes                      : 151176339
Total Size of DS in bytes               : 15095376
Number of DSRs within the DS            : 2241
Length of the DSRs in bytes             : 6736


DATA SET DESCRIPTOR #20:
Data Set Name                           : MERIS_SOURCE_PACKETS
Data Set Type                           : R
External    Product    Reference                                                :
MER_FR__0PNPDE20030921_092217_000002582020_00079_08149_1418.N1
DS Offset in bytes                      : 0
Total Size of DS in bytes               : 0
Number of DSRs within the DS            : 0
Length of the DSRs in bytes             : 0


DATA SET DESCRIPTOR #21:
Data Set Name                           : INSTRUMENT_DATA_FILE
Data Set Type                           : R
External    Product    Reference                                                :
MER_INS_AXVIEC20030620_120000_20020321_193100_20121008_190821
DS Offset in bytes                      : 0
Total Size of DS in bytes               : 0
Number of DSRs within the DS            : 0
Length of the DSRs in bytes             : 0


DATA SET DESCRIPTOR #22:
Data Set Name                           : PROCESSING_PARAMS_L1B_FILE
Data Set Type                           : R
External    Product    Reference                                                :
MER_CP1_AXVIEC20030620_120000_20020429_040000_20120920_173421
DS Offset in bytes                      : 0
Total Size of DS in bytes               : 0
Number of DSRs within the DS            : 0
Length of the DSRs in bytes             : 0


DATA SET DESCRIPTOR #23:
Data Set Name                           : RADIOMETRIC_CALIBRATION_FILE
Data Set Type                           : R
External    Product    Reference                                                :
MER_RAC_AXVIEC20030620_120000_20021224_121445_20121224_121445
DS Offset in bytes                      : 0
Total Size of DS in bytes               : 0
Number of DSRs within the DS            : 0
Length of the DSRs in bytes             : 0


DATA SET DESCRIPTOR #24:
Data Set Name                           : DIGITAL_ELEVATION_MODEL_FILE
Data Set Type                           : R
External    Product    Reference                                                :
AUX_DEM_AXVIEC20020123_121901_20020101_000000_20200101_000000
DS Offset in bytes                      : 0
Total Size of DS in bytes               : 0
Number of DSRs within the DS            : 0
Length of the DSRs in bytes             : 0


DATA SET DESCRIPTOR #25:
Data Set Name                           : DIGITAL_ROUGHNESS_MODEL_FILE
Data Set Type                           : R
```

```
External    Product    Reference                                                    :
MER_DRM_AXVIEC20020122_083343_20020101_000000_20200101_000000
DS Offset in bytes                      : 0
Total Size of DS in bytes               : 0
Number of DSRs within the DS            : 0
Length of the DSRs in bytes             : 0


DATA SET DESCRIPTOR #26:
Data Set Name                           : LAND_SEA_MASK_DATA_FILE
Data Set Type                           : R
External    Product    Reference                                                    :
AUX_LSM_AXVIEC20020123_141228_20020101_000000_20200101_000000
DS Offset in bytes                      : 0
Total Size of DS in bytes               : 0
Number of DSRs within the DS            : 0
Length of the DSRs in bytes             : 0


DATA SET DESCRIPTOR #27:
Data Set Name                           : ECMWF_DATA_FILE
Data Set Type                           : R
External    Product    Reference                                                    :
AUX_ECA_AXNECM20030921_233653_20030921_090000_20030921_210000
DS Offset in bytes                      : 0
Total Size of DS in bytes               : 0
Number of DSRs within the DS            : 0
Length of the DSRs in bytes             : 0


DATA SET DESCRIPTOR #28:
Data Set Name                           : ORBIT_STATE_VECTOR_FILE
Data Set Type                           : R
External    Product    Reference                                                    :
AUX_FRO_AXVPDS20030924_101146_20030920_221000_20030923_005000
DS Offset in bytes                      : 0
Total Size of DS in bytes               : 0
Number of DSRs within the DS            : 0
Length of the DSRs in bytes             : 0


DATA SET DESCRIPTOR #29:
Data Set Name                           : ATTITUDE_DATA_FILE
Data Set Type                           : R
External    Product    Reference                                                    :
AUX_ATT_AXVIEC20020924_131534_20020703_120000_20781231_235959
DS Offset in bytes                      : 0
Total Size of DS in bytes               : 0
Number of DSRs within the DS            : 0
Length of the DSRs in bytes             : 0


DATA SET #1 - "Quality ADS":
```

No.	Date	Blank	Out of Range 1	Out of Range 2	Out of Range 3	Out of Range 4	Out of Range 5	Out R. blank 1	Out R. blank 2	Out R. blank 3	Out R. blank 4	Out R. blank 5
1	117451421.714409	0	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
2	117451444.240873	0	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
3	117451466.767337	0	.....	.....	.....	.....	.....	.....	.....	.....	1.....	.....
4	117451489.293801	0	.....	.....	.....	.....	.....	.....	.....	.....	1.....	.....
5	117451511.820265	0	.....	.....	.....	.....	.....	.....	.....	.....	1.....	.....

```
DATA SET #2 - "Scaling Factor GADS":
Scaling factor - altitude               : 1.000000
Scaling factor - roughness              : 1.000000
Scaling factor - zonal wind             : 0.100000
Scaling factor - meridional wind        : 0.100000
Scaling factor - atmospheric pressure   : 0.100000
Scaling factor - ozone                  : 0.010000
Scaling factor - relative humidity      : 0.100000
Radiances:
   Band 1                               : 0.009333
   Band 2                               : 0.010466
   Band 3                               : 0.011501
   Band 4                               : 0.010642
   Band 5                               : 0.009280
   Band 6                               : 0.008147
```

```
   Band 7                              : 0.006815
   Band 8                              : 0.006899
   Band 9                              : 0.006270
   Band 10                             : 0.008617
   Band 11                             : 0.008811
   Band 12                             : 0.003613
   Band 13                             : 0.003546
   Band 14                             : 0.006090
   Band 15                             : 0.005418
Gain setting:
   Module #1 - Band 1                  : 9
   Module #1 - Band 2                  : 10
   Module #1 - Band 3                  : 11
   Module #1 - Band 4                  : 11
   Module #1 - Band 5                  : 11
   Module #1 - Band 6                  : 11
   Module #1 - Band 7                  : 11
   Module #1 - Band 8                  : 10
   Module #1 - Band 9                  : 11
   Module #1 - Band 10                 : 11
   Module #1 - Band 11                 : 11
   Module #1 - Band 12                 : 10
   Module #1 - Band 13                 : 9
   Module #1 - Band 14                 : 11
   Module #1 - Band 15                 : 10
   Module #1 - Band 16                 : 0
   Module #2 - Band 1                  : 9
   Module #2 - Band 2                  : 10
   Module #2 - Band 3                  : 11
   Module #2 - Band 4                  : 11
   Module #2 - Band 5                  : 11
   Module #2 - Band 6                  : 11
   Module #2 - Band 7                  : 11
   Module #2 - Band 8                  : 10
   Module #2 - Band 9                  : 11
   Module #2 - Band 10                 : 11
   Module #2 - Band 11                 : 11
   Module #2 - Band 12                 : 10
   Module #2 - Band 13                 : 9
   Module #2 - Band 14                 : 11
   Module #2 - Band 15                 : 10
   Module #2 - Band 16                 : 0
   Module #3 - Band 1                  : 9
   Module #3 - Band 2                  : 10
   Module #3 - Band 3                  : 11
   Module #3 - Band 4                  : 11
   Module #3 - Band 5                  : 11
   Module #3 - Band 6                  : 11
   Module #3 - Band 7                  : 11
   Module #3 - Band 8                  : 10
   Module #3 - Band 9                  : 11
   Module #3 - Band 10                 : 11
   Module #3 - Band 11                 : 11
   Module #3 - Band 12                 : 10
   Module #3 - Band 13                 : 9
   Module #3 - Band 14                 : 11
   Module #3 - Band 15                 : 10
   Module #3 - Band 16                 : 0
   Module #4 - Band 1                  : 9
   Module #4 - Band 2                  : 10
   Module #4 - Band 3                  : 11
   Module #4 - Band 4                  : 11
   Module #4 - Band 5                  : 11
   Module #4 - Band 6                  : 11
   Module #4 - Band 7                  : 11
   Module #4 - Band 8                  : 10
   Module #4 - Band 9                  : 11
   Module #4 - Band 10                 : 11
   Module #4 - Band 11                 : 11
   Module #4 - Band 12                 : 10
   Module #4 - Band 13                 : 9
   Module #4 - Band 14                 : 11
   Module #4 - Band 15                 : 10
```

```
   Module #4 - Band 16                  : 0
   Module #5 - Band 1                   : 9
   Module #5 - Band 2                   : 10
   Module #5 - Band 3                   : 11
   Module #5 - Band 4                   : 11
   Module #5 - Band 5                   : 11
   Module #5 - Band 6                   : 11
   Module #5 - Band 7                   : 11
   Module #5 - Band 8                   : 10
   Module #5 - Band 9                   : 11
   Module #5 - Band 10                  : 11
   Module #5 - Band 11                  : 11
   Module #5 - Band 12                  : 10
   Module #5 - Band 13                  : 9
   Module #5 - Band 14                  : 11
   Module #5 - Band 15                  : 10
   Module #5 - Band 16                  : 0
Sampling rate                           : 43997 (1.e-6 s)
Sun spectral flux:
   Band 1                               : 1703.201782
   Band 2                               : 1859.746582
   Band 3                               : 1911.582886
   Band 4                               : 1914.315063
   Band 5                               : 1789.023315
   Band 6                               : 1637.997314
   Band 7                               : 1519.083252
   Band 8                               : 1459.818726
   Band 9                               : 1395.926514
   Band 10                              : 1255.756104
   Band 11                              : 1243.889160
   Band 12                              : 1167.314209
   Band 13                              : 950.546509
   Band 14                              : 922.478821
   Band 15                              : 888.092041
```

DATA SET #3 - "Tie points ADS":

No.	Date	FIRST (Lat., Lon.)	Alt. Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt. Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	117451421.714409	(45.909863, 18.014693)	98 3	0 1e-6	0 1e-6 (48.2, 153.6)	(0.0, 182.0)	(47.028196, 10.603496)	1476 382	-1907 1e-6	-16512 1e-6 (51.8, 145.0)	(40.7, 99.6)
9	117451444.240873	(44.594348, 17.519381)	591 159	0 1e-6	0 1e-6 (47.2, 152.5)	(0.0, 181.9)	(45.703992, 10.278711)	850 227	-1101 1e-6	-9284 1e-6 (50.8, 144.1)	(40.7, 99.6)
17	117451466.767337	(43.277165, 17.041762)	417 32	0 1e-6	0 1e-6 (46.1, 151.4)	(0.0, 181.9)	(44.379209, 9.960598)	744 75	-967 1e-6	-7942 1e-6 (49.8, 143.2)	(40.7, 99.7)
25	117451489.293801	(41.958432, 16.580328)	-110 0	0 1e-6	0 1e-6 (45.1, 150.3)	(0.0, 181.9)	(43.053869, 9.648478)	-366 0	0 1e-6	0 1e-6 (48.9, 142.3)	(40.7, 99.7)
33	117451511.820265	(40.638252, 16.133722)	540 177	0 1e-6	0 1e-6 (44.1, 149.2)	(0.0, 181.8)	(41.727992, 9.341743)	116 23	-152 1e-6	-1186 1e-6 (47.9, 141.3)	(40.7, 99.7)
36	117451520.267689	(40.142832, 15.969815)	1053 209	0 1e-6	0 1e-6 (43.7, 148.8)	(0.0, 181.8)	(41.230654, 9.227991)	79 54	-103 1e-6	-801 1e-6 (47.6, 140.9)	(40.7, 99.8)

APPENDIX B - ORTHORECTIFICATION LOGS

This section contains the log of MERSYN execution when applied to MERIS FR scene MER_FR__1PNUPA20030921_092341_000000982020_00079_08149_0534.N1 of Italy.

These logs contain in particular information regarding:

- Automatic settings – Size, origin and ground sampling automatically computed by MERSYN.
- Prediction/correction loop perfoances – Loop convergence tracing for point (117,872) and convergence statistics,
- Runtime performances,
- Statistics on the number of processed points.

B.1 No orthorectification

```
SETTING PIXEL SIZE DEFAULT VALUES
   PIXEL HEIGHT = 260.000000
   PIXEL WIDTH  = 260.000000

COMPUTING IMAGE SIZE AND ORIGIN
   UL: X = 1025533.187500   Y = 5226455.019000
   LR: X = 2002024.500000   Y = 4461188.500000
   IMAGE HEIGHT = 765267.519000 metres
   IMAGE WIDTH  = 976491.312500 metres
   LINE  NUMBER = 2943
   PIXEL NUMBER = 3755
LINE=  106  elapsed/line=       0  facet search: -same= 98.8%  -neighbour=  1.0%  -all=
0.2%
INITIALISATION:              INVERSE:       lonlat_origin=(0.196665,0.816031)       -->
mer_xy_origin(72.692636,2028.052061)  facet=(1,31)  delta=(0.135822,0.688313)
LINE= 2942  elapsed/line=       0  facet search: -same= 98.7%  -neighbour=  1.3%  -all=
0.0%

Total elapsed time=27.700 seconds  elapsed time/facet=15.630 seconds ( 56.4%)


PREDICTION/CORRECTION LOOP STATISTICS  (total processed points = 7818186)
. ITERATION [ 0] =    7818186  (100.0 %)
. ITERATION [ 1] =          0  ( 0.0 %)
. ITERATION [ 2] =          0  ( 0.0 %)
. ITERATION [ 3] =          0  ( 0.0 %)
. ITERATION [ 4] =          0  ( 0.0 %)
. ITERATION [ 5] =          0  ( 0.0 %)
. ITERATION [ 6] =          0  ( 0.0 %)
. ITERATION [ 7] =          0  ( 0.0 %)
. ITERATION [ 8] =          0  ( 0.0 %)
. ITERATION [ 9] =          0  ( 0.0 %)
. ITERATION [10] =          0  ( 0.0 %)

ORTHORECTIFICATION PROCESSING         = NO
NUMBER OF PROCESSED POINTS IN SEGMENT = 7815250  (70.7% of synthesis)
NUMBER  OF  VALIDATED  POINTS  IN  SEGMENT  =  7815250   (100.0%  of  segment)  (70.7%  of
synthesis)
```

B.2 Altitudes from tie-point grid

```
SETTING PIXEL SIZE DEFAULT VALUES
   PIXEL HEIGHT = 260.000000
   PIXEL WIDTH  = 260.000000

COMPUTING IMAGE SIZE AND ORIGIN
   UL: X = 1025533.187500   Y = 5226455.019000
```

```
   LR: X = 2002024.500000   Y = 4461188.500000
   IMAGE HEIGHT = 765267.519000 metres
   IMAGE WIDTH  = 976491.312500 metres
   LINE  NUMBER = 2943
   PIXEL NUMBER = 3755
LINE=  106  elapsed/line=   20000  facet search: -same= 88.8%  -neighbour= 10.0%  -all=
1.2%
INITIALISATION:              INVERSE:       lonlat_origin=(0.196665,0.816031)       -->
mer_xy_origin(72.692636,2028.052061)  facet=(1,31)  delta=(0.135822,0.688313)

PREDICTION_CORRECTION_NUMBER=0
   dem_altitude=1406.003360
      Interpolated lonlat=(0.196665,0.816031)
      Viewing zenith=0.657754   Viewing azimuth=1.747770
      dlon=-0.000245   dlat=-0.000030   meters=(-1560.593089,-191.218894)
   DIRECT: mer_xy0=(72.692636,2028.052061) --> lonlat1=(0.196910,0.816061)   dlonlat=(-
0.000245,-0.000030) = (-1560.591599,-191.216959)
   INVERSE:      lonlat1=(0.196910,0.816061)      -->      mer_xy1=(71.400363,2024.121496)
facet=(1,31)  delta=(0.115631,0.626898)
   NEW FACET for corrected point (73.984910,2031.982627):   iline/ipixel_facet=(1,31)
delta_facet_xy=(0.156014,0.749729)

PREDICTION_CORRECTION_NUMBER=1
   dem_altitude=1347.322220
      Interpolated lonlat=(0.196420,0.816001)
      Viewing zenith=0.658753   Viewing azimuth=1.747602
      dlon=-0.000235   dlat=-0.000029   meters=(-1498.548552,-183.444896)
   DIRECT:      mer_xy0=(73.984910,2031.982627)      -->      lonlat1=(0.196655,0.816030)
dlonlat=(0.000010,0.000001) = (61.864355,7.774689)
   INVERSE:      lonlat1=(0.196655,0.816030)      -->      mer_xy1=(72.744521,2028.207753)
facet=(1,31)  delta=(0.136633,0.690746)
   NEW FACET for corrected point (73.933025,2031.826935):   iline/ipixel_facet=(1,31)
delta_facet_xy=(0.155204,0.747296)

PREDICTION_CORRECTION_NUMBER=2
   dem_altitude=1349.637452
      Interpolated lonlat=(0.196430,0.816003)
      Viewing zenith=0.658713   Viewing azimuth=1.747609
      dlon=-0.000236   dlat=-0.000029   meters=(-1501.001059,-183.751925)
   DIRECT: mer_xy0=(73.933025,2031.826935) --> lonlat1=(0.196665,0.816031)   dlonlat=(-
0.000000,-0.000000) = (-2.445607,-0.307057)
   INVERSE:      lonlat1=(0.196665,0.816031)      -->      mer_xy1=(72.690586,2028.045906)
facet=(1,31)  delta=(0.135790,0.688217)
   NEW FACET for corrected point (73.935075,2031.833090):   iline/ipixel_facet=(1,31)
delta_facet_xy=(0.155236,0.747392)
LINE= 2942  elapsed/line=       0  facet search: -same= 97.9%  -neighbour=  2.1%  -all=
0.0%

Total elapsed time=68.760 seconds  elapsed time/facet=41.250 seconds ( 60.0%)


PREDICTION/CORRECTION LOOP STATISTICS  (total processed points = 7818186)
. ITERATION [ 0] =       2936  ( 0.0 %)
. ITERATION [ 1] =    2852732  (36.5 %)
. ITERATION [ 2] =    4833499  (61.8 %)
. ITERATION [ 3] =     129019  ( 1.7 %)
. ITERATION [ 4] =          0  ( 0.0 %)
. ITERATION [ 5] =          0  ( 0.0 %)
. ITERATION [ 6] =          0  ( 0.0 %)
. ITERATION [ 7] =          0  ( 0.0 %)
. ITERATION [ 8] =          0  ( 0.0 %)
. ITERATION [ 9] =          0  ( 0.0 %)
. ITERATION [10] =          0  ( 0.0 %)

ORTHORECTIFICATION PROCESSING         = YES
NUMBER OF PROCESSED POINTS IN SEGMENT = 7806229  (70.6% of synthesis)
NUMBER  OF  VALIDATED  POINTS  IN  SEGMENT  =  7806229   (100.0%  of  segment)  (70.6%  of
synthesis)
```

B.3 DEM Italy

```
SETTING PIXEL SIZE DEFAULT VALUES
   PIXEL HEIGHT = 260.000000
   PIXEL WIDTH  = 260.000000

COMPUTING IMAGE SIZE AND ORIGIN
   UL: X = 1025533.187500   Y = 5226455.019000
   LR: X = 2002024.500000   Y = 4461188.500000
   IMAGE HEIGHT = 765267.519000 metres
   IMAGE WIDTH  = 976491.312500 metres
   LINE  NUMBER = 2943
   PIXEL NUMBER = 3755
LINE=  106  elapsed/line=   50000  facet search: -same= 88.7%  -neighbour= 10.0%  -all=
1.3%
INITIALISATION:              INVERSE:       lonlat_origin=(0.196665,0.816031)       -->
mer_xy_origin(72.692636,2028.052061)  facet=(1,31)  delta=(0.135822,0.688313)

PREDICTION_CORRECTION_NUMBER=0
   dem_altitude=2230.704269
      Interpolated lonlat=(0.196665,0.816031)
      Viewing zenith=0.657754   Viewing azimuth=1.747770
      dlon=-0.000389   dlat=-0.000048   meters=(-2475.969664,-303.379646)
   DIRECT: mer_xy0=(72.692636,2028.052061) --> lonlat1=(0.197054,0.816079)   dlonlat=(-
0.000389,-0.000048) = (-2475.968174,-303.377712)
   INVERSE:      lonlat1=(0.197054,0.816079)      -->      mer_xy1=(70.642402,2021.816213)
facet=(1,31)  delta=(0.103788,0.590878)
   NEW FACET for corrected point (74.742871,2034.287910):   iline/ipixel_facet=(1,31)
delta_facet_xy=(0.167857,0.785749)

PREDICTION_CORRECTION_NUMBER=1
   dem_altitude=2230.704269
      Interpolated lonlat=(0.196276,0.815984)
      Viewing zenith=0.659339   Viewing azimuth=1.747504
      dlon=-0.000390   dlat=-0.000048   meters=(-2484.086078,-303.922619)
   DIRECT: mer_xy0=(74.742871,2034.287910) --> lonlat1=(0.196666,0.816032)   dlonlat=(-
0.000001,-0.000000) = (-8.570086,-0.541638)
   INVERSE:      lonlat1=(0.196666,0.816032)      -->      mer_xy1=(72.687253,2028.030131)
facet=(1,31)  delta=(0.135738,0.687971)
   NEW FACET for corrected point (74.748255,2034.309840):   iline/ipixel_facet=(1,31)
delta_facet_xy=(0.167941,0.786091)
LINE= 2942  elapsed/line=       0  facet search: -same= 97.9%  -neighbour=  2.0%  -all=
0.0%

Total elapsed time=325.960 seconds  elapsed time/facet=44.140 seconds ( 13.5%)


PREDICTION/CORRECTION LOOP STATISTICS  (total processed points = 7818186)
. ITERATION [ 0] =       2936  ( 0.0 %)
. ITERATION [ 1] =    3008252  (38.5 %)
. ITERATION [ 2] =    4798944  (61.4 %)
. ITERATION [ 3] =       8054  ( 0.1 %)
. ITERATION [ 4] =          0  ( 0.0 %)
. ITERATION [ 5] =          0  ( 0.0 %)
. ITERATION [ 6] =          0  ( 0.0 %)
. ITERATION [ 7] =          0  ( 0.0 %)
. ITERATION [ 8] =          0  ( 0.0 %)
. ITERATION [ 9] =          0  ( 0.0 %)
. ITERATION [10] =          0  ( 0.0 %)

ORTHORECTIFICATION PROCESSING         = YES
NUMBER OF PROCESSED POINTS IN SEGMENT = 7806166  (70.6% of synthesis)
NUMBER  OF  VALIDATED  POINTS  IN  SEGMENT  =  7806166   (100.0%  of  segment)  (70.6%  of
synthesis)
```

B.4 DEM SRTM

```
SETTING PIXEL SIZE DEFAULT VALUES
   PIXEL HEIGHT = 260.000000
   PIXEL WIDTH  = 260.000000

COMPUTING IMAGE SIZE AND ORIGIN
   UL: X = 1025533.187500   Y = 5226455.019000
   LR: X = 2002024.500000   Y = 4461188.500000
   IMAGE HEIGHT = 765267.519000 metres
   IMAGE WIDTH  = 976491.312500 metres
   LINE  NUMBER = 2943
   PIXEL NUMBER = 3755
LINE=  106  elapsed/line=   40000  facet search: -same= 88.8%  -neighbour=  9.8%  -all=
1.4%
INITIALISATION:              INVERSE:       lonlat_origin=(0.196665,0.816031)       -->
mer_xy_origin(72.692636,2028.052061)  facet=(1,31)  delta=(0.135822,0.688313)

PREDICTION_CORRECTION_NUMBER=0
   dem_altitude=2106.815331
      Interpolated lonlat=(0.196665,0.816031)
      Viewing zenith=0.657754   Viewing azimuth=1.747770
      dlon=-0.000367   dlat=-0.000045   meters=(-2338.459168,-286.530536)
   DIRECT: mer_xy0=(72.692636,2028.052061) --> lonlat1=(0.197032,0.816076)   dlonlat=(-
0.000367,-0.000045) = (-2338.457677,-286.528601)
   INVERSE:      lonlat1=(0.197032,0.816076)      -->      mer_xy1=(70.756263,2022.162509)
facet=(1,31)  delta=(0.105567,0.596289)
   NEW FACET for corrected point (74.629009,2033.941614):   iline/ipixel_facet=(1,31)
delta_facet_xy=(0.166078,0.780338)

PREDICTION_CORRECTION_NUMBER=1
   dem_altitude=2106.815331
      Interpolated lonlat=(0.196298,0.815986)
      Viewing zenith=0.659251   Viewing azimuth=1.747519
      dlon=-0.000368   dlat=-0.000045   meters=(-2345.698628,-287.014884)
   DIRECT: mer_xy0=(74.629009,2033.941614) --> lonlat1=(0.196666,0.816032)   dlonlat=(-
0.000001,-0.000000) = (-7.644129,-0.483121)
   INVERSE:      lonlat1=(0.196666,0.816032)      -->      mer_xy1=(72.687834,2028.032501)
facet=(1,31)  delta=(0.135747,0.688008)
   NEW FACET for corrected point (74.633811,2033.961175):   iline/ipixel_facet=(1,31)
delta_facet_xy=(0.166153,0.780643)
LINE= 2942  elapsed/line=       0  facet search: -same= 97.9%  -neighbour=  2.0%  -all=
0.0%

Total elapsed time=304.440 seconds  elapsed time/facet=43.130 seconds ( 14.2%)


PREDICTION/CORRECTION LOOP STATISTICS  (total processed points = 7818186)
. ITERATION [ 0] =       2936  ( 0.0 %)
. ITERATION [ 1] =    3034335  (38.8 %)
. ITERATION [ 2] =    4777361  (61.1 %)
. ITERATION [ 3] =       3554  ( 0.0 %)
. ITERATION [ 4] =          0  ( 0.0 %)
. ITERATION [ 5] =          0  ( 0.0 %)
. ITERATION [ 6] =          0  ( 0.0 %)
. ITERATION [ 7] =          0  ( 0.0 %)
. ITERATION [ 8] =          0  ( 0.0 %)
. ITERATION [ 9] =          0  ( 0.0 %)
. ITERATION [10] =          0  ( 0.0 %)

ORTHORECTIFICATION PROCESSING         = YES
NUMBER OF PROCESSED POINTS IN SEGMENT = 7806080  (70.6% of synthesis)
NUMBER  OF  VALIDATED  POINTS  IN  SEGMENT  =  7806080   (100.0%  of  segment)  (70.6%  of
synthesis)
```

B.5 DEM GETASSE30

```
MERSYN -mer ../scene01-04/MER_FR__1PNUPA20030921_092341_000000982020_00079_08149_0534.N1
-osy synthesis_dem-getasse.{3,5,7} -dem ../DEM_GETASSE30/EUROPA_WGS84 -sam "Nearest
neighbour" -prj "projection=Plate_carree ellipsoid=WGS_1984 datum=WGS_84" -cha 3 5 7

SETTING PIXEL SIZE DEFAULT VALUES
   PIXEL HEIGHT = 260.000000
   PIXEL WIDTH  = 260.000000

COMPUTING IMAGE SIZE AND ORIGIN
   UL: X = 1025533.187500   Y = 5226455.019000
   LR: X = 2002024.500000   Y = 4461188.500000
   IMAGE HEIGHT = 765267.519000 metres
   IMAGE WIDTH  = 976491.312500 metres
   LINE  NUMBER = 2943
   PIXEL NUMBER = 3755
LINE= 2942  elapsed/line=       0  facet search: -same= 98.4%  -neighbour=  1.5%  -all=
0.0%

Total elapsed time=317.750 seconds  elapsed time/facet=42.450 seconds ( 13.4%)


PREDICTION/CORRECTION LOOP STATISTICS  (total processed points = 7818186)
. ITERATION [ 0] =       2936  ( 0.0 %)
. ITERATION [ 1] =    2811159  (36.0 %)
. ITERATION [ 2] =    5004091  (64.0 %)
. ITERATION [ 3] =          0  ( 0.0 %)
. ITERATION [ 4] =          0  ( 0.0 %)
. ITERATION [ 5] =          0  ( 0.0 %)
. ITERATION [ 6] =          0  ( 0.0 %)
. ITERATION [ 7] =          0  ( 0.0 %)
. ITERATION [ 8] =          0  ( 0.0 %)
. ITERATION [ 9] =          0  ( 0.0 %)
. ITERATION [10] =          0  ( 0.0 %)

ORTHORECTIFICATION PROCESSING         = YES
NUMBER OF PROCESSED POINTS IN SEGMENT = 7810559  (70.7% of synthesis)
NUMBER OF VALIDATED POINTS IN SEGMENT = 7810559  (100.0% of segment) (70.7% of
synthesis)

BENCHMAK POINTS
##  LINE PIXEL    GCP LON.     GCP LAT.  GCP ALT.   EASTING    NORTHING     LONGITUDE
LATITUDE ALTITUDE  VIEW.ZENITH VIEW.AZIMUTH   LON.CORR.   LAT.CORR.   ORIGIN: LINE PIXEL
MERIS: LINE PIXEL
 0    38  788     11.071550    46.939982  2864    1230413.2   5216575.0     11.071550
46.939981  2841       38.825      99.946       0.029666   -0.003551     12.362 2098.622
12.375 2107.442
 1   149  968     11.492668    46.680293  2054    1277213.2   5187715.0     11.492668
46.680292  2036       36.737     100.298       0.019604   -0.002443     90.297 1957.374
90.302 1963.237
 2   204 1431     12.575876    46.551618  1229    1397593.2   5173415.0     12.575875
46.551617  1424       31.672     101.089       0.011273   -0.001519     85.682 1633.586
85.684 1636.974
 3   231 1792     13.420450    46.488450  1265    1491453.2   5166395.0     13.420450
46.488449  1436       27.462     101.695       0.009548   -0.001361     65.485 1383.903
65.488 1386.782
 4   728 1972     13.841567    45.325698   112    1538253.2   5037175.0     13.841568
45.325697   164       23.385     102.166       0.000884   -0.000134    473.198 1158.107
473.198 1158.380
 5   397 2487     15.046431    46.100086   944    1672153.2   5123235.0     15.046432
46.100085   985       18.091     102.895       0.004068   -0.000646    118.239  877.737
118.240  878.979
 6   784 2745     15.650033    45.194683   230    1739233.2   5022615.0     15.650033
45.194683   252       12.812     103.431       0.000710   -0.000119    414.628  613.340
414.628  613.561
 7   507 3234     16.794069    45.842737   158    1866373.2   5094635.0     16.794069
45.842736   158        7.285     104.152       0.000252   -0.000044    105.166  345.520
105.166  345.598
 8  1308 2745     15.650033    43.968764   210    1739233.2   4886375.0     15.650033
43.968763   215       10.309     103.573       0.000474   -0.000082    864.013  490.957
864.013  491.107
```

```
 9   916 3260     16.854897    44.885864   194    1873133.2   4988295.0     16.854897
44.885863   196        4.804     104.297       0.000203   -0.000037    450.513  227.214
450.513  227.278
10  1303 3312     16.976553    43.980461  1551    1886653.2   4887675.0     16.976554
43.980461  1516        2.016     104.478       0.000646   -0.000120    772.409   94.983
772.409   95.189
11   419  942     11.431840    46.048616   460    1270453.2   5117515.0     11.431840
46.048616   522       36.219     100.377       0.004872   -0.000619    329.537 1926.552
329.536 1928.026
12   806  994     11.553496    45.143214    46    1283973.2   5016895.0     11.553496
45.143213    47       34.585     100.619       0.000402   -0.000053    661.958 1820.076
661.958 1820.200
13  1247  525     10.456251    44.111476   359    1162033.2   4902235.0     10.456251
44.111475   459       38.335     100.049       0.004474   -0.000569   1101.176 2070.374
1101.176 2071.773
14  1827  473     10.334595    42.754542    48    1148513.2   4751435.0     10.334594
42.754541    48       37.390     100.208       0.000442   -0.000058   1614.970 2005.610
1614.970 2005.751
15  1275 1014     11.600287    44.045968   502    1289173.2   4894955.0     11.600287
44.045968   559       33.026     100.830       0.004465   -0.000614   1068.898 1718.696
1068.900 1720.097
16  1827  937     11.420142    42.754541   582    1269153.2   4751435.0     11.420142
42.754541   588       32.249     100.918       0.004465   -0.000632   1559.338 1669.616
1559.338 1671.048
17  1523 1529     12.805151    43.465762   510    1423073.2   4830475.0     12.805151
43.465761   473       26.044     101.727       0.002805   -0.000423   1219.772 1304.177
1219.772 1305.069
18  1993 1478     12.685834    42.366178   544    1409813.2   4708275.0     12.685834
42.366177   583       24.994     101.805       0.003241   -0.000500   1633.713 1244.989
1633.714 1246.038
19  2407 1761     13.347924    41.397607   646    1483393.2   4600635.0     13.347924
41.397607   637       19.533     102.362       0.002647   -0.000435   1952.375  951.629
1952.377  952.501
20  2158 2096     14.131670    41.980153   919    1570493.2   4665375.0     14.131671
41.980152   919       15.805     102.796       0.003069   -0.000518   1689.644  760.930
1689.646  761.933
21  2241 2869     15.940136    41.785971   617    1771473.2   4643795.0     15.940137
41.785970   662        3.816     103.997       0.000517   -0.000096   1642.937  180.101
1642.937  180.271
22  2517 2328     14.674444    41.140258   243    1630813.2   4572035.0     14.674444
41.140257   304       10.719     103.244       0.000669   -0.000119   1963.413  510.395
1963.413  510.617
23  2683 2843     15.879308    40.751894   756    1764713.2   4528875.0     15.879308
40.751893   780        1.953     104.063       0.000306   -0.000058   2024.333   92.061
2024.333   92.164
24   944  654     10.758052    44.820357    64    1195573.2   4981015.0     10.758052
44.820356    66       37.769     100.131       0.000634   -0.000080    821.305 2032.557
821.305 2032.753
```

APPENDIX C – MERSYN SOURCE CODE

```
/*****************************************************************************/
/* NAME                                                                      */
/* MERSYN (abbreviation "syn") adds the MERIS segment to the synthesis.      */
/*****************************************************************************/
/* SYNOPSIS                                                                  */
/* MERSYN [<sequence_name>] [-mer <MERIS_segment_in_input>]                  */
/*                          [-isy <input_synthesis_channel_list>]            */
/*                          [-ivi <input_NDVI_maximum_image>]                */
/*                          [-osy <output_synthesis_channel_list>]           */
/*                          [-ovi <output_NDVI_maximum_image>]               */
/*                          [-sal <selection_algorithm>]                     */
/*                          [-dem <input_DEM_list>]                          */
/*                          [-sam <sampling_method>]                         */
/*                          [-prj <Projection>]                              */
/*                          [-cha <Channel identification>]                  */
/*                          [-ul  <upper-left corner coordinates>]           */
/*                          [-lr  <lower-right corner coordinates>]          */
/*                          [-pxh <pixel_height>]                            */
/*                          [-pxw <pixel_width>]                             */
/*                          [-ort <process_orthorectification>]              */
/*****************************************************************************/
/* DESCRIPTION                                                               */
/* MERSYN merges radiances found within the MERIS segment into the various   */
/* bands of the synthesis provided in input. Merging is performed according to*/
/* an algorithm to avoid clouds and invalid values.                          */
/*                                                                           */
/* MERIS products in input of MERSYN may be any scene or segments 1B with the */
/* full or reduced resolution.                                               */
/*                                                                           */
/* Products in output of MERSYN are orthorectified. I.e., the parallax defect */
/* due to the relief and the view angle are corrected. At each point of the  */
/* output synthesis, the elevation value may be interpolated from the        */
/* elevation data found within the "tie-point grid" or retrieved from one or */
/* more DEMs given as "-dem" parameter.                                      */
/* At each point of the output synthesis, the viewing angle is interpolated  */
/* from the values found in the "tie-point grid".                            */
/*                                                                           */
/*                                                                           */
/* MERSYN may be used in three different contexts. In each one of these cases */
/* the set of files in input and output differ according to different         */
/* parameter values.                                                         */
/*                                                                           */
/* USE CASE 1 - Minimum-radiance synthesis                                   */
/* ---------------------------------------                                   */
/*                                                                           */
/*   1x  -mer      Nx  -isy       Dx  -dem                                   */
/*       MERIS         input           DEM                                   */
/*      segment      synthesis         list                                   */
/*          \           |              .                                      */
/*           \          |             .                                       */
/*            \         |            .                                        */
/*                    MERSYN                  Required parameters            */
/*                       |                    -sal = "Radiance minimum"      */
/*                       |                                                   */
/*                       |                                                   */
/*                  Nx  -osy                                                 */
/*                     output                                                */
/*                     synthesis                                             */
/*                                                                           */
/* In this use case 1, the selection algorithm (parameter "-sal") has the    */
/* value "Radiance minimum". The N bands of the output synthesis are computed */
/* from the N bands of the input synthesis in which the corresponding pixels */
/* of the MERIS segment replaces the "synthesis pixel" if its radiance is    */
/* inferior and if the "MERIS pixel" is not flagged as corrupted.            */
/*                                                                           */
/*                                                                           */
/* USE CASE 2 - Maximum-NDVI synthesis                                       */
/* -----------------------------------                                       */
/*                                                                           */
/*   1x  -mer      Nx  -isy     1x  -ivi     Dx  -dem                       */
/*       MERIS         input        input         DEM                       */
/*      segment      synthesis       NDVI         list                      */
/*          \           |            /            .                        */
/*           \          |           /            .                         */
/*            \         |          /           .                          */
/*                    MERSYN                  Required parameters          */
/*                       |          \         -sal = "NDVI maximum"         */
/*                       |           \                                      */
/*                       |            \                                     */
/*                  Nx  -osy     1x  -ovi                                    */
/*                     output       output                                   */
/*                     synthesis       NDVI                                  */
/*                                                                           */
/* In this use case 2, the selection algorithm (parameter "-sal") has the    */
/* value "NDVI maximum". The N bands of the output synthesis are computed    */
/* from the N bands of the input synthesis in which the corresponding pixels */
/* of all the N bands of the MERIS segment replaces the "synthesis pixels" of */
/* the N bands if the NDVI computed from the red and NIR bands is greater    */
/* than the one found in the "input NDVI image".                             */
/*                                                                           */
/*                                                                           */
/* USE CASE 3 - Mapping                                                      */
/* --------------------                                                      */
/*                                                                           */
/*   1x  -mer                      Dx  -dem                                  */
/*       MERIS                         DEM                                 */
/*      segment                        list                                 */
/*          \                           .                                   */
/*           \                         .                                    */
/*            \                       .                                     */
/*                    MERSYN                  Required parameters          */
/*                       |                    -cha  selected bands         */
/*                       |                    -prj  output projection      */
/*                       |                    -ul   upper-left coordinates */
/*                  Nx  -osy                  -lr   lower-right coordinates */
/*                     output                 -pxh  pixel height           */
/*                     synthesis              -pxw  pixel width            */
/*                                                                           */
/* In this use case 3, MERSYN acts as a simple orthorectification process    */
/* (like MAPPER) producing N bands (parameter "-cha") of geocoded images which*/
/* projection ("-prj"), bounding box ("-ul","-lr") and pixel sizel ("-pxh",  */
/* "-pxw") are defined by the user and not implicitely deduced from the input */
/* synthesis like in the other cases.                                        */
/*                                                                           */
/* The selection algorithm is assumed to have the "Radiance minimum" value.  */
/* This use case 3, may be used as the first step to produce a first synthesis*/
/* which will be iteratively upgraded like in the use cases 1 and 2.         */
/* Parameters "-ul" and "-lr" enables clipping a particular area of interest. */
/*                                                                           */
/* When no values are given by user for these parameters, the bounding box   */
/* will be automatically computed by MERSYN as the smallest rectangle that   */
/* includes the entire MERIS segment in input.                               */
/*                                                                           */
/* When no values are given by user for pixel size parameters ("-pxh" and    */
/* "-pxw"), MERSYN will set these parameters to the nominal value of the FR or*/
/* RR products.                                                              */
/*                                                                           */
/* Unlike the other use cases, when no synthesis is provided in input, MERSYN */
/* will no chek if the pixel is corrupted looking for the values found in    */
/* MDS(16). All the pixels of the image will be mapped.                      */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/* PARAMETERS                                                                */
/* ----------                                                                */
/* Input segment (<MERIS_segment_in_input>) is a Foreign file formatted in   */
/* Earth Explorer (ENVISAT) format. Its name should normally start with      */
/* "MER_RR__1P..." prefix and should have ".N1" as suffix. Decoding of its   */
/* Main Product Header (MHS) and Specific Product Header (SPH) will lead to  */
/* a long listing printed on the standard output.                            */
/*                                                                           */
/* Synthesis in input includes a list of 16-bits 2D-Raster files matching the */
/* various bands of the MERIS data to be processed. MERIS instrument has 15  */
/* bands in visible and near infrared spectra. The "channel" ressource of the */
/* provided channels shall be distinct and within the range [1,15].          */
/* Synthesis images includes a background to indicate that no valid value has */
/* been yet found for this pixel location.                                   */
/*                                                                           */
/* When the "NDVI maximum" algorithm is selected, the max. NDVI matching the */
/* initialised values shall be kept and provided as <input_NDVI_maximum_image>*/
/* parameter. For the first time (i.e. if the synthesis does not yet include */
/* any segment), this "-ivi" parameter may not be given.                     */
/* NDVI values are in the range [-1,+1] and are represented within a 16-bits */
/* image within the range [1,65535] (value 0 being to represent the          */
/* background).                                                              */
/*                                                                           */
/* Synthesis in output (<output_synthesis_channel_list>) contains a file list */
/* matching the synthesis provided in input (same number of bands and channel */
/* list). For each pixel, if the pixel is declared as valid, pixels of all the*/
/* bands will be updated according to the algorithm.                         */
/*                                                                           */
/* The NDVI maximum image (<output_NDVI_maximum_image>) is an updated version */
/* of the <input_NDVI_maximum_image> in which the pixels having a NDVI greater*/
/* than the one found in input (if any) will be set to the value of this     */
/* maximum.                                                                  */
/* When no NDVI maximum image is provided in input, all the pixels of the    */
/* output NDVI image will be initialised with a background value.            */
/*****************************************************************************/
/* DOCUMENT                                                                  */
/* "PO-RS-MDA-GS-2009 - ENVISAT PAYLOAD DATA SEGMENT - ENVISAT-1 PRODUCTS    */
/*  SPECIFICATIONS - VOLUME 11: MERIS PRODUCTS SPECIFICATIONS - Issue 3      */
/*  Revision F - 20/11/2000 - ESA, ALCATEL SPACE"                            */
/*****************************************************************************/
/* COPYRIGHTS                                                                */
/* Copyrights 2003 by GAEL Consultant - All rights Reserved.                 */
/* GAEL Consultant Proprietary - Delivered under License Agreement.          */
/* Copying and Disclosure Prohibited Without Express Written Permission From */
/* GAEL Consultant.                                                          */
/*****************************************************************************/
/* ADMINISTRATION                                                            */
/* Serge RIAZANOFF  | 11.05.03 | v00.01 | Creation of the SW component        */
/* Serge RIAZANOFF  | 15.05.03 | v00.02 | Maximum NDVI option               */
/* Serge RIAZANOFF  | 29.05.03 | v00.03 | Flag is_replaced -> added_point_nu..*/
/* Stephane MBAYE   | 09.08.03 | v00.04 | Checked/Addapted for FR mode      */
/* Serge RIAZANOFF  | 04.11.03 | v01.01 | Case of MAPPER behaviour          */
/* Serge RIAZANOFF  | 25.12.03 | v01.02 | Management of large files ( > 2GB ) */
/* Serge RIAZANOFF  | 12.07.08 | v02.01 | Full new release                  */
/* Serge RIAZANOFF  | 09.08.04 | v02.02 | Fixing of prediction/correction bugs*/
/* Serge RIAZANOFF  | 15.08.04 | v02.03 | "-ort" option added               */
/* Serge RIAZANOFF  | 07.11.04 | v02.04 | #ifdef TIM_BENCH  (Generat.bench pt)*/
/* Serge RIAZANOFF  | 03.01.05 | v02.05 | Change in dlat dlon corrections   */
/*****************************************************************************/

/*****************************************************************************/
/* ANSI C inclusion files                                                    */
/*****************************************************************************/
#include     <stdio.h>
#include     <stdlib.h>
#include     <string.h>
#include     <math.h>
#include     <memory.h>
#include     <errno.h>
#ifdef TIM_PERF
#include     <time.h>
#endif

/*---------------------------------------------------------------------------*/
/* X inclusion files are required while using TimFmt (progress bar)          */
/*---------------------------------------------------------------------------*/
#include     <X11/Intrinsic.h>
#include     <Xm/Xm.h>

/*****************************************************************************/
/* TELIMAGO inclusion files                                                  */
/*****************************************************************************/
#define _TimNeedProcess
#define _TimNeedFile
#define _TimNeedFormat
#define _TimNeedPrj

#include     <Tim.h>
#include     <TimX.h>

/*****************************************************************************/
/* Global constants (maximum 31 characters)                                  */
/*****************************************************************************/
#define MODULE_NAME "STATION"          /* module including this process*/
/*---------------------------------------------------------------------------*/
/* TIM_MERSYN_BREAK_DETECTION is a large maximum of the possible X-difference */
/* between adjacent vertices of the segment boundary. When two successive    */
/* points have a difference greater than this value, a -180/+180 break is    */
/* assumed.                                                                  */
/*---------------------------------------------------------------------------*/
#define TIM_MERSYN_BREAK_DETECTION                      (90 / 180. * TIM_PI)

#define TIM_MERSYN_PREDICTION_CORRECTION_MAX_ITERATION   10

#define TIM_MERSYN_SELECTION_ALGORITHM_RADIANCE_MINIMUM  0
```

```
#define TIM_MERSYN_SELECTION_ALGORITHM_NDVI_MAXIMUM      1

#define TIM_MERSYN_MERIS_TYPE_FR                         0
#define TIM_MERSYN_MERIS_TYPE_RR                         1

#define TIM_MERSYN_PARALLELISM_THRESHOLD                 ((double)1.e-12)

#define TIM_MERSYN_HOT_AREA                              (TIM_PI-.01)

#define TIM_MERSYN_MEAN_EARTH_RADIUS                     6370997.0

/*--------------------------------------------------------------------------*/
/* Precision expressed in part of output pixel size                         */
/*--------------------------------------------------------------------------*/
#define TIM_MERSYN_MAX_ERROR                             0.1

/*--------------------------------------------------------------------------*/
/* Number of lines per channels (selected channels + NDVI(if any) + MDS(16))  */
/* To optimize access, many lines are kept within the buffer that is        */
/* permutated modulo the number of the line to access.                      */
/*--------------------------------------------------------------------------*/
#define TIM_MERIS_BUFFER_MAX_LINE_NUMBER                 1000


/****************************************************************************/
/* Type definitions                                                         */
/****************************************************************************/
/* Input parameters structure                                               */
/*--------------------------------------------------------------------------*/
typedef struct {
   TimRForeign      mer;               /* MERIS segment in input */
   TimR2DRasterList isy;               /* Input synthesis channel list */
   TimR2DRaster     ivi;               /* Input NDVI maximum image */
   TimR2DRasterList osy;               /* Output synthesis channel list */
   TimR2DRaster     ovi;               /* Output NDVI maximum image */
   TimRString       sal;               /* Selection algorithm */
   TimRDEMList      dem;               /* input dem list */
   TimRString       sam;               /* Sampling method */
   TimRString       prj;               /* Projection */
   TimRIntegerList  cha;               /* Channel identification */
   TimRVector       ul;                /* Output image upper left corner */
   TimRVector       lr;                /* Output image lower right corner */
   TimRFloat        pxh;               /* Output image pixel height */
   TimRFloat        pxw;               /* Output image pixel width */
   TimRBoolean      ort;               /* Process orthorectification */
} TimMersynIp;                         /* PROCESS PARAMETERS */
/*--------------------------------------------------------------------------*/
/* Buffer used to access and keep MERIS data                                */
/*--------------------------------------------------------------------------*/
typedef struct {
   unsigned short   lineIndex [TIM_MERIS_BUFFER_MAX_LINE_NUMBER];
                                       /* line index of data stored in array */
   unsigned short   *line     [TIM_MERIS_BUFFER_MAX_LINE_NUMBER];
                                       /* image line */
} TimMerisBuffer;                      /* PROCESS PARAMETERS */


/****************************************************************************/
/* Forward declarations                                                     */
/****************************************************************************/
/*--------------------------------------------------------------------------*/
int TimMersynComputeBoundary TIM_PROTO((
TimRForeign         meris_file,         /* MERIS segment in input */
TimPrjProjection    *geographic_projection, /* projection of tie-points */
TimPrjProjection    *synthesis_projection,  /* output image projection */
TimRPolyVectorList  *boundary_p         /* enveloppe to be computed */
));
/*--------------------------------------------------------------------------*/
int TimMersynComputeLineIntersection TIM_PROTO((
double              point0_x,           /* X-coordinate of point 1 of D1 */
double              point0_y,           /* Y-coordinate of point 1 of D1 */
double              point1_x,           /* X-coordinate of point 2 of D1 */
double              point1_y,           /* Y-coordinate of point 2 of D1 */
double              point2_x,           /* X-coordinate of point 1 of D2 */
double              point2_y,           /* Y-coordinate of point 1 of D2 */
double              point3_x,           /* X-coordinate of point 2 of D2 */
double              point3_y,           /* Y-coordinate of point 2 of D2 */
double              *point_x_p,         /* X-coordinate of concurrence point */
double              *point_y_p          /* Y-coordinate of concurrence point */
));
/*--------------------------------------------------------------------------*/
int TimMersynComputeIntersectionsWithBoundary TIM_PROTO((
double              y_current,          /* curr.point Y-coord in output proje.*/
TimRPolyVectorList  *boundary_p,        /* enveloppe of MERIS segment */
TimRFloatList       *intersection_p,    /* list of intersect. segment % line */
int                 *is_inside_p,       /* "point is inside segment" flag */
int                 *current_intersection_p  /* intersection to be reached */
));
/*--------------------------------------------------------------------------*/
int TimMersynGetAltitudeFromDem TIM_PROTO((
double              longitude,          /* X-coord of point to be processed */
double              latitude,           /* Y-coord of point to be processed */
TimPrjProjection    *geographic_projection, /* projection structure */
TimRDEMList         *dem_p,             /* DEMs in order they have to be read */
double              *altitude_p         /* altitude to be initialized */
));
/*--------------------------------------------------------------------------*/
int TimMersynIsInFacet TIM_PROTO((
int                 iline_facet,        /* line index of the UL tie-pt corner */
int                 ipixel_facet,       /* pixel index of the UL tie-pt corner*/
double              syn_lon, /* X-coord. point in synthesis */
double              syn_lat, /* X-coord. point in synthesis */
double              *delta_facet_x,     /* line-position to compute */
double              *delta_facet_y      /* pixel-position to compute */
));
/*--------------------------------------------------------------------------*/
int TimMersynLocationDirect TIM_PROTO((
double              mer_x,              /* X-coordinates in MERIS segment */
double              mer_y,              /* Y-coordinates in MERIS segment */
double              altitude,           /* Z-coordinates in MERIS segment */
int                 iline_facet,        /* number of the tie-point frame */
int                 ipixel_facet,       /* number of the tie-point column */
double              delta_facet_x,      /* X-coordinates within facet [0,1] */
double              delta_facet_y,      /* Y-coordinates within facet [0,1] */
double              *longitude_p,       /* longitude to be computed */
double              *latitude_p         /* latitude to be computed */
));
/*--------------------------------------------------------------------------*/
int TimMersynLocationInverse TIM_PROTO((
double              longitude,          /* X-coord of point to be processed */
double              latitude,           /* Y-coord of point to be processed */
int                 *iline_facet_p,     /* number of the tie-point frame */
int                 *ipixel_facet_p,    /* number of the tie-point column */
double              *delta_facet_x_p,   /* X-coordinates within facet [0,1] */
double              *delta_facet_y_p,   /* Y-coordinates within facet [0,1] */
double              *mer_x_p,           /* X-coordinates in MERIS segment */
double              *mer_y_p            /* Y-coordinates in MERIS segment */
));
/*--------------------------------------------------------------------------*/
int TimMersynReadLine TIM_PROTO((
TimRForeign         mer,                /* MERIS segment in input */
int                 iband,              /* spectral band to be processed */
int                 iline,              /* line to be read */
TimMerisBuffer      *meris_buffer_p     /* MERIS buffer to initialize */
));
/*--------------------------------------------------------------------------*/


/****************************************************************************/
/* Global variables                                                         */
/****************************************************************************/
int                 process_MERIS_flag = TIM_TRUE;
int                 channel_index_red  = 6;  /* BAND  7 in MERIS buffer list */
int                 channel_index_NIR  = 12; /* BAND 13 in MERIS buffer list */
/*--------------------------------------------------------------------------*/
/* Used to monitor performance                                              */
/*--------------------------------------------------------------------------*/
#ifdef TIM_PERF
clock_t             current_clock;      /* current clock value */
long                time_line0;         /* clock value of the 1st line */
long                time_line;          /* current clock value of the line */
long                time_facet;         /* clock value of facet research start*/
long                elapsed_time_facet; /* elapsed time of facet search / line*/
int                 facet_search_same      = 0;  /* found within current */
int                 facet_search_neighbour = 0;  /* found around current */
int                 facet_search_all       = 0;  /* search within all facets */
#endif

/*--------------------------------------------------------------------------*/
/* Used to print out bench points                                           */
/*--------------------------------------------------------------------------*/
#ifdef TIM_BENCH
struct {
   TimR2DCoordinates gcp;               /* GCP collected in GEOREF application*/
   double             easting;          /* synthesis horizontal coordinate */
   double             northing;         /* synthesis vertical coordinate */
   double             longitude;        /* geodetic longitude */
   double             latitude;         /* geodetic latitude */
   double             altitude;         /* interpolated altitude */
   double             viewingZenith;    /* latest viewing zenith */
   double             viewingAzimuth;   /* latest viewing azimuth */
   double             longitudeCorrection; /* latest longithde correction */
   double             latitudeCorrection;  /* latest latithde correction */
   double             merisX0;          /* X-coord in MERIS segment, 1st iter.*/
   double             merisY0;          /* Y-coord in MERIS segment, 1st iter.*/
   double             merisX;           /* X-coord in MERIS segment,last iter.*/
   double             merisY;           /* Y-coord in MERIS segment,last iter.*/
} TIM_MERSYN_BENCH[] = {
{{   38,  788, 8.19257241855421148e-01, 1.93235007893579031e-01, 2.8640e+03 }},
{{  149,  968, 8.14724808475713269e-01, 2.00584894386977269e-01, 2.0540e+03 }},
{{  204, 1431, 8.12479007695734001e-01, 2.19490434571149889e-01, 1.2290e+03 }},
{{  231, 1792, 8.11376523852257003e-01, 2.34231039082843873e-01, 1.2650e+03 }},
{{  728, 1972, 7.91082662090661137e-01, 2.41580924142774267e-01, 1.1200e+02 }},
{{  397, 2487, 8.04598292212129973e-01, 2.62609766892887175e-01, 9.4400e+02 }},
{{  784, 2745, 7.88796031052489188e-01, 2.73144604332492325e-01, 2.3000e+02 }},
{{  507, 3234, 8.00106694851500877e-01, 2.93111802997920723e-01, 1.5800e+02 }},
{{ 1308, 2745, 7.67399691433426345e-01, 2.73144603481369597e-01, 2.1000e+02 }},
{{  916, 3260, 7.83406114503204076e-01, 2.94173452210109931e-01, 1.9400e+02 }},
{{ 1303, 3312, 7.67603856633425452e-01, 2.96296753023537141e-01, 1.5510e+03 }},
{{  419,  942, 8.03699972987327871e-01, 1.99523244202737482e-01, 4.6000e+02 }},
{{  806,  994, 7.87897711395466605e-01, 2.01646544668149574e-01, 4.6000e+01 }},
{{ 1247,  525, 7.69890487863912787e-01, 1.82496007596055609e-01, 3.5900e+02 }},
{{ 1827,  473, 7.46207520406638847e-01, 1.80372706900750301e-01, 4.8000e+01 }},
{{ 1275, 1014, 7.68747170708711125e-01, 2.02463198669260630e-01, 5.0200e+02 }},
{{ 1827,  937, 7.46207519009472464e-01, 1.99319080792183273e-01, 5.8200e+02 }},
{{ 1523, 1529, 7.58620659256733609e-01, 2.23492038218818967e-01, 5.1000e+02 }},
{{ 1993, 1478, 7.39429289943063561e-01, 2.21409570773334397e-01, 5.4400e+02 }},
{{ 2407, 1761, 7.22524552646203966e-01, 2.32965224840718077e-01, 6.4600e+02 }},
{{ 2158, 2096, 7.32691894508517461e-01, 2.46644178905278511e-01, 9.1900e+02 }},
{{ 2241, 2869, 7.29302781755709395e-01, 2.78207860731241208e-01, 6.1700e+02 }},
{{ 2517, 2328, 7.18032956127631738e-01, 2.56117366684067871e-01, 2.4300e+02 }},
{{ 2683, 2843, 7.11254729211615366e-01, 2.77146211985425106e-01, 7.5600e+02 }},
{{  944,  654, 7.82262798066772791e-01, 1.87763426261147615e-01, 6.4000e+01 }}
};
   int              bench_index;        /* index of current bench point */
   int              ibench;             /* index among bench points */
   double           null_elevation_longitude; /* used to compute correction */
   double           null_elevation_latitude;  /* used to compute correction */
#endif


/****************************************************************************/
/* main                                                                     */
/****************************************************************************/
int main
#ifdef TIM_ANSI_PROTOTYPE
(
int                 argc,               /* argument count */
char                **argv              /* argument values */
)
#else
 (argc,argv)
int                 argc;               /* argument count */
char                **argv;             /* argument values */
#endif
{
/****************************************************************************/
/* Local variables (maximum 31 characters)                                  */
/****************************************************************************/
   TimMersynIp      ip;                 /* input parameters structure */
   TimProcess       process;            /* standard process structure */
   TimRInteger      sampling_method;    /* integer value of sampling method */
   int              selection_algorithm;/* numerical value of ip.sal */
   int              process_ortho;      /* process orthorectification */
   int              meris_type;         /* numerical product type (FR or RR) */
   TimRSensor       header;             /* MERIS segment header */
   TimMerisBuffer   *i_buf_meris[16];   /* MERIS buffers (radiances+ndvi+flag)*/
   TimBufferInt     **i_buf_synthesis=NULL;/* input buffer list for synthesis */
   TimBufferInt     **o_buf_synthesis=NULL;/* output buffer list for synthesis*/
   TimBufferFloat   *i_buf_ndvi=NULL;   /* input buffer for NDVI maximum */
   TimBufferFloat   *o_buf_ndvi=NULL;   /* output buffer for NDVI maximum */
   TimFmtBufferList fmt_buffer;         /* list of buffers to access TimFmt */
   TimRFileList     input_file;         /* struct used to pass ip.mer */
   TimPrjProjection *geographic_projection=NULL; /* projection of tie-points */
   TimPrjProjection *synthesis_projection=NULL;  /* output image projection */
   TimPrjCoordinate prj_coordinate;     /* convertion prj coordinate struct */
   TimRPolyVectorList boundary;         /* coord. of the segment enveloppe(s) */
#ifdef TIM_PERF
   int              prediction_correction_histo[
                       TIM_MERSYN_PREDICTION_CORRECTION_MAX_ITERATION+1];
                                        /* prediction/correction histogram */
   int              ihisto;             /* index among items in histogram */
   int              point_number;       /* total number of samples in histo. */
#endif

   double           syn_lon_origin;     /* longitude of current point in synth*/
   double           syn_lat_origin;     /* latitude  of current point in synth*/
   double           syn_lon1;           /* longitude of (mer_x0,mer_y0) image */
   double           syn_lat1;           /* latitude  of (mer_x0,mer_y0) image */
```

```
   double           mer_x_origin;       /* 1st antecedent X-coord */
   double           mer_y_origin;       /* 1st antecedent Y-coord */
   double           mer_x0;             /* previous antecedent X-coord */
   double           mer_y0;             /* previous antecedent Y-coord */
   double           mer_x1;             /* current antecedent X-coord */
   double           mer_y1;             /* current antecedent Y-coord */

   int              band_number;        /* number of bands in the synthesis */
   int              iband;              /* index among bands in list */
   int              iline;              /* index among lines in image */
   int              ipixel;             /* index among columns in image */
   int              iline_facet;        /* line index of current tie-pt facet */
   int              ipixel_facet;       /* pixel index of current tie-pt facet*/
   int              iboundary;          /* index among boundaries in list */
   int              ipoint;             /* index among points in list */
   int              iline_facet_first; /* memory of last full search: line */
   int              ipixel_facet_first; /* memory of last full search: pixel */
   int              iline_in_buffer;    /* nb of the line within MERIS buffer */
   int              i;                  /* convenient index */
   int              iline_segment;      /* NN int. of origin pixel in segment */
   int              ipixel_segment;     /* NN int. of origin pixel in segment */
   TimRFloatList    intersection;       /* list of intersect. segment % line */
   int              current_intersection; /* intersection to be reached */
   double           delta_facet_x;      /* line-position within facet */
   double           delta_facet_y;      /* pixel-position within facet */
   int              prediction_correction_number; /* number of iterations */
   double           dem_altitude;       /* local altitude of DEM from facet */
   int              processed_point_number; /* stat. checked pts in segment */
   int              added_point_number; /* stat. valid pts in segment */
   TimRDouble       image_height;       /* height of output image */
   TimRDouble       image_width;        /* height of output image */
   double           x_min;              /* used to compute bounding box */
   double           x_max;              /* used to compute bounding box */
   double           y_min;              /* used to compute bounding box */
   double           y_max;              /* used to compute bounding box */
   double           x_current;          /* curr.point X-coord in output proje.*/
   double           y_current;          /* curr.point Y-coord in output proje.*/
   unsigned short   flag_value;         /* value of current pixel flags */
   int              channel_index;      /* absolute channel index: 0->15 */
   int              radiance_old;       /* previous value of synthes.radiance */
   int              radiance_new;       /* new value of synthesis radiance */
   int              radiance_new_red;   /* new value of synthes. red radiance */
   int              radiance_new_NIR;   /* new value of synthes. NIR radiance */
   double           ndvi_old;           /* previous value of NDVI */
   double           ndvi_new;           /* new value of NDVI */
   int              ndvi_min_pixel;     /* min. value in NDVI file represent. */
   int              ndvi_max_pixel;     /* max. value in NDVI file represent. */
   double           ndvi_min_value;     /* min. value in external NDVI repres.*/
   double           ndvi_max_value;     /* max. value in external NDVI repres.*/
   int              pixel_value;        /* auxiliary variable for int. value */

   int              status;             /* status returned by I/O functions */
   int              is_inside;          /* "point is inside segment" flag */
   int              is_found;           /* "facet has been found" flag */
   int              done;               /* "loop is completed" flag */
   int              altitude_found;     /* "altitude has been found" flag */
   int              process_pixel;      /* "pixel may be processed" flag */
   int              is_background;      /* "current pixel is background" flag */
   int              is_replaced;        /* "at least one band replaced" flag */
   int              is_first_of_line;   /* "Is the 1st pixel processed" flag */


/*****************************************************************************/
/* Initialize application                                                    */
/*****************************************************************************/
   if (TimInit(&argc,argv,MODULE_NAME,&process,&ip) != TIM_OK)
      TimExit (&process,TIM_PRS_PARAM);
/*****************************************************************************/
/* Initialize files                                                          */
/*****************************************************************************/
   if (TimInitProcessing(&process,TIM_FILE_ALL) != TIM_OK)
      TimExit (&process,TIM_PRS_ERROR);
   if (ip.isy.num > 0)
      band_number = ip.isy.num;
   else
      band_number = ip.cha.num;
/*****************************************************************************/
/* Check parameters                                                          */
/*****************************************************************************/
/* Check algorithm options                                                   */
/*---------------------------------------------------------------------------*/
   if      (strcmp(ip.sal,"Radiance minimum") == 0)
      selection_algorithm = TIM_MERSYN_SELECTION_ALGORITHM_RADIANCE_MINIMUM;
   else if (strcmp(ip.sal,"NDVI maximum") == 0)
      selection_algorithm = TIM_MERSYN_SELECTION_ALGORITHM_NDVI_MAXIMUM;
   else
   {
      fprintf(TimStderr,"MERSYN: Invalid \"-sal\" parameter.\n");
      TimExit (&process,TIM_PRS_PARAM);
   }
/*---------------------------------------------------------------------------*/
/* Check synthesis files                                                     */
/*---------------------------------------------------------------------------*/
   if (ip.isy.num > 0)
   {
/*---------------------------------------------------------------------------*/
/*    Check that all the channels have the same required ressources          */
/*---------------------------------------------------------------------------*/
      for (iband=0; iband<band_number; iband++)
      {
         if ((ip.isy.list[iband]->lineNumber  != ip.isy.list[0]->lineNumber)  ||
             (ip.isy.list[iband]->pixelNumber != ip.isy.list[0]->pixelNumber) ||
             (ip.isy.list[iband]->dataDepth   != 16)                          ||
             (ip.isy.list[iband]->backgroundFlag != TIM_TRUE)                 ||
             (ip.isy.list[iband]->background  != 0))
         {
            fprintf(TimStderr,
               "MERSYN: Ressources of input synthesis images are not valid.\n");
            TimExit (&process,TIM_PRS_PARAM);
         }
      }
      for (iband=0; iband<band_number; iband++)
      {
         if ((ip.osy.list[iband]->lineNumber  != ip.osy.list[0]->lineNumber)  ||
             (ip.osy.list[iband]->pixelNumber != ip.osy.list[0]->pixelNumber) ||
             (ip.osy.list[iband]->dataDepth   != 16)                          ||
             (ip.osy.list[iband]->backgroundFlag != TIM_TRUE)                 ||
             (ip.osy.list[iband]->background  != 0))
         {
            fprintf(TimStderr,
               "MERSYN: Ressources of output synthesis images are not valid.\n");
            TimExit (&process,TIM_PRS_PARAM);
         }
      }
/*---------------------------------------------------------------------------*/
/*    Check that all the channels are geocoded with the same projection      */
/*    ressources                                                             */
/*---------------------------------------------------------------------------*/
      for (iband=0; iband<band_number; iband++)
      {
         if ((ip.isy.list[iband]->projection == NULL)                        ||
             (strcmp(ip.isy.list[iband]->projection,
                     ip.isy.list[0]->projection) != 0)                       ||
             (ip.isy.list[iband]->origin.x != ip.isy.list[0]->origin.x)      ||
             (ip.isy.list[iband]->origin.y != ip.isy.list[0]->origin.y)      ||
             (ip.isy.list[iband]->pixelHeight <= 0.0)                        ||
             (ip.isy.list[iband]->pixelHeight != ip.isy.list[0]->pixelHeight) ||
             (ip.isy.list[iband]->pixelWidth <= 0.0)                         ||
             (ip.isy.list[iband]->pixelWidth != ip.isy.list[0]->pixelWidth))
         {
            fprintf(TimStderr,"MERSYN: Invalid projection ressources.\n");
            TimExit (&process,TIM_PRS_PARAM);
         }
      }
   }
/*---------------------------------------------------------------------------*/
/* Check the presence of NDVI file                                           */
/*---------------------------------------------------------------------------*/
   if (selection_algorithm == TIM_MERSYN_SELECTION_ALGORITHM_NDVI_MAXIMUM)
   {
      if (ip.ovi == NULL)
      {
         fprintf(TimStderr,"MERSYN: Algorithm based on NDVI analysis => "
            "Output NDVI file is required.\n");
         TimExit (&process,TIM_PRS_PARAM);
      }
   }
   else
   {
      if (ip.ovi != NULL)
      {
         fprintf(TimStderr,
            "MERSYN: Warning: Algorithm not based on NDVI analysis => "
            "Output NDVI file is not required.\n");
      }
   }
/*===========================================================================*/
/* Analyze the sampling method                                               */
/*===========================================================================*/
   if      (strcmp(ip.sam,TIM_SAMPLING_NEAREST_NEIGHBOUR) == 0)
      sampling_method = TIM_SAMPLING_METHOD_NEAREST_NEIGHBOUR;
   else if (strcmp(ip.sam,TIM_SAMPLING_BI_LINEAR) == 0)
      sampling_method = TIM_SAMPLING_METHOD_BI_LINEAR;
   else if (strcmp(ip.sam,TIM_SAMPLING_BI_CUBIC) == 0)
      sampling_method = TIM_SAMPLING_METHOD_BI_CUBIC;
   else if (strcmp(ip.sam,TIM_SAMPLING_SINC) == 0)
      sampling_method = TIM_SAMPLING_METHOD_SINC;
   else if (strcmp(ip.sam,TIM_SAMPLING_MEAN) == 0)
      sampling_method = TIM_SAMPLING_METHOD_MEAN;
   else
   {
      fprintf (TimStderr,
        "MERSYN: Warning: Unrecognized sampling method \"%s\". "
        "Setting default one: \"Nearest neighbour\"\n",ip.sam);
      sampling_method = TIM_SAMPLING_METHOD_NEAREST_NEIGHBOUR;
   }
/*===========================================================================*/
/* When input synthesis is provided, the "-prj", "-cha", "-ul", "-lr", "-pxh" */
/* nor "-pxw" parameters shall not be provided by user                       */
/*===========================================================================*/
   if (ip.isy.num != 0)
   {
      if ((ip.prj != NULL)                                                   ||
          (ip.cha.num != 0)                                                  ||
          ((ip.ul.x != 0.0) || (ip.ul.y != 0.0))                             ||
          ((ip.lr.x != 0.0) || (ip.lr.y != 0.0))                             ||
          (ip.pxh != 0.0)                                                    ||
          (ip.pxw != 0.0))
      {
         fprintf(TimStderr,
           "MERSYN: When the input synthesis is provided,\"-prj\", \"-cha\", "
            "\"-ul\", \"-lr\", \"-pxh\" nor \"-pxw\" parameters shall be "
            "provided.\n");
         TimExit (&process,TIM_PRS_PARAM);
      }
   }
/*===========================================================================*/
/* When input synthesis is not provided, the "-prj", "-cha", "-ul", "-pxh" and*/
/* "-pxw" parameters shall all be provided by user                           */
/*===========================================================================*/
   else
   {
/*---------------------------------------------------------------------------*/
/*    Check that this parameters have been initialised                       */
/*---------------------------------------------------------------------------*/
      if ((ip.prj == NULL)                                                   ||
          (ip.cha.num == 0))
/* Automatic setting of default values. See here after
         ((ip.ul.x == 0.0) && (ip.ul.y == 0.0))                              ||
         ((ip.lr.x == 0.0) && (ip.lr.y == 0.0))                              ||
         (ip.pxh == 0.0)                                                     ||
         (ip.pxw == 0.0)
*/
      {
         fprintf(TimStderr,
            "MERSYN: When the input synthesis is not provided, "
            "\"-prj\", \"-cha\" "
/*
            ", \"-ul\", \"-lr\", \"-pxh\" and \"-pxw\" "
*/
            "parameters shall all be provided.\n");
         TimExit (&process,TIM_PRS_PARAM);
      }
/*---------------------------------------------------------------------------*/
/*    Channel number must match the number of channels in output synthesis   */
/*---------------------------------------------------------------------------*/
      if (ip.cha.num != ip.osy.num)
      {
         fprintf(TimStderr,
            "MERSYN: When the channel identification list is provided, "
            "its number of items (%d) shall exactly match the number of bands "
            "(%d) of the output synthesis \"-osy\".\n",ip.cha.num,ip.osy.num);
         TimExit (&process,TIM_PRS_PARAM);
      }
/*---------------------------------------------------------------------------*/
/*    Check the UL / LR coordinates                                          */
/*---------------------------------------------------------------------------*/
      if ((ip.ul.x > ip.lr.x)                                                ||
          (ip.ul.y < ip.lr.y))
      {
         fprintf(TimStderr,"MERSYN: Inconsistant UL / LR values.\n");
         TimExit (&process,TIM_PRS_PARAM);
      }
/*---------------------------------------------------------------------------*/
/*    Check the pixel size                                                   */
/*---------------------------------------------------------------------------*/
      if ((ip.pxh < 0.0)                                                     ||
          (ip.pxw < 0.0))
      {
         fprintf(TimStderr,"MERSYN: Invalid pixel height or width.\n");
         TimExit (&process,TIM_PRS_PARAM);
```

```
    }
  }
/*=========================================================================*/
/* When input synthesis is not provided, the "-ivi" parameter shall not be */
/* provided                                                                */
/*=========================================================================*/
  if (ip.isy.num == 0)
  {
    if (ip.ivi != NULL)
    {
      fprintf(TimStderr,
        "MERSYN: When the input synthesis is not provided,\"-ivi\" "
        "parameter shall not be provided.\n");
      TimExit (&process,TIM_PRS_PARAM);
    }
  }
/*=========================================================================*/
/* When input synthesis is not provided, the MDS(16) flags will not be     */
/* processed                                                               */
/*=========================================================================*/
  if (ip.isy.num == 0)
  {
    process_MERIS_flag = TIM_FALSE;
  }
/*=========================================================================*/
/* When no orthorectification is required, no DEM shall be provided in input */
/*=========================================================================*/
  if (!(ip.ort))
  {
    if (ip.dem.num != 0)
    {
      fprintf(TimStderr,"MERSYN: When no orthorectification is required, "
        "no DEM shall be provided in input.\n");
      TimExit (&process,TIM_PRS_PARAM);
    }
  }
  process_ortho = ip.ort;
/*************************************************************************/
/* Read MERIS segment header                                             */
/*************************************************************************/
  header            = NULL;
  fmt_buffer.num    = 0;
  fmt_buffer.list   = NULL;
  input_file.num    = 1;
  input_file.list   = (TimRFile*)TimAlloc(NULL,1,sizeof(TimRFile));
  input_file.list[0] = (TimRFile)(ip.mer);
  if ((status=TimFmtEnvisatMerisHeaderRead(&input_file,TIM_FMT_LEVEL_STANDARD,
      &fmt_buffer,TIM_FALSE,(TimRFile*)(&header))) == TIM_FMT_ERROR)
  {
    fprintf(TimStderr,"MERSYN: Error while decoding \"%s\".\n",
      ip.mer->name);
    TimExit (&process,TIM_PRS_ERROR);
  }
  if (status != TIM_FMT_VALID)
  {
    fprintf(TimStderr,
      "MERSYN: MERIS format has not been recognized in \"%s\".\n",
      ip.mer->name);
    TimExit (&process,TIM_PRS_ERROR);
  }
/*-------------------------------------------------------------------------*/
/* Determine the type of MERIS segment to be processed                     */
/*-------------------------------------------------------------------------*/
  if (strncmp(&(TIM_FMT_ENVISAT_MERIS_SPECIAL.SphDescriptor[4]),"FR",2) == 0)
    meris_type = TIM_MERSYN_MERIS_TYPE_FR;
  else
  if (strncmp(&(TIM_FMT_ENVISAT_MERIS_SPECIAL.SphDescriptor[4]),"RR",2) == 0)
    meris_type = TIM_MERSYN_MERIS_TYPE_RR;
  else
  {
    fprintf(TimStderr,
      "MERSYN: Unknown MERIS type. SPH descriptor=\"%s\".\n",
      TIM_FMT_ENVISAT_MERIS_SPECIAL.SphDescriptor);
    TimExit (&process,TIM_PRS_PARAM);
  }
/*************************************************************************/
/* Set output file ressources                                            */
/*************************************************************************/
/* When the input synthesis is not provided, all the resources shall be set */
/* here (no heritage from "-isy")                                          */
/*-------------------------------------------------------------------------*/
  if (ip.isy.num == 0)
  {
    for (iband=0; iband<ip.osy.num; iband++)
    {
      ip.osy.list[iband]->dataDepth        = 16;
      ip.osy.list[iband]->backgroundFlag   = TIM_TRUE;
      ip.osy.list[iband]->background       = TIM_BACKGROUND;
      ip.osy.list[iband]->projection       = TimStrDup(ip.prj);
/*-------------------------------------------------------------------------*/
/*       Pixel height                                                      */
/*-------------------------------------------------------------------------*/
      if (ip.pxh != 0.0)
      {
        ip.osy.list[iband]->pixelHeight = ip.pxh;
      }
      else
      {
        if (strstr(ip.osy.list[iband]->projection,"projection=Geographic")
            != NULL)
        {
          if (meris_type == TIM_MERSYN_MERIS_TYPE_RR)
          {
            ip.osy.list[iband]->pixelHeight  =
              TIM_FMT_ENVISAT_MERIS_RR_PIXEL_SIZE /
              TIM_MERSYN_MEAN_EARTH_RADIUS;
          }
          else if (meris_type == TIM_MERSYN_MERIS_TYPE_FR)
          {
            ip.osy.list[iband]->pixelHeight  =
              TIM_FMT_ENVISAT_MERIS_FR_PIXEL_SIZE /
              TIM_MERSYN_MEAN_EARTH_RADIUS;
          }
        }
        else
        {
          if (meris_type == TIM_MERSYN_MERIS_TYPE_RR)
          {
            ip.osy.list[iband]->pixelHeight  =
              TIM_FMT_ENVISAT_MERIS_RR_PIXEL_SIZE;
          }
          else if (meris_type == TIM_MERSYN_MERIS_TYPE_FR)
          {
            ip.osy.list[iband]->pixelHeight  =
              TIM_FMT_ENVISAT_MERIS_FR_PIXEL_SIZE;
          }
        }
      }
/*-------------------------------------------------------------------------*/
/*       Pixel width                                                       */
/*-------------------------------------------------------------------------*/
      if (ip.pxw != 0.0)
      {
        ip.osy.list[iband]->pixelWidth = ip.pxw;
      }
      else
      {
        if (strstr(ip.osy.list[iband]->projection,"projection=Geographic")
            != NULL)
        {
          if (meris_type == TIM_MERSYN_MERIS_TYPE_RR)
          {
            ip.osy.list[iband]->pixelWidth  =
              TIM_FMT_ENVISAT_MERIS_RR_PIXEL_SIZE /
              TIM_MERSYN_MEAN_EARTH_RADIUS;
          }
          else if (meris_type == TIM_MERSYN_MERIS_TYPE_FR)
          {
            ip.osy.list[iband]->pixelWidth  =
              TIM_FMT_ENVISAT_MERIS_FR_PIXEL_SIZE /
              TIM_MERSYN_MEAN_EARTH_RADIUS;
          }
        }
        else
        {
          if (meris_type == TIM_MERSYN_MERIS_TYPE_RR)
          {
            ip.osy.list[iband]->pixelWidth  =
              TIM_FMT_ENVISAT_MERIS_RR_PIXEL_SIZE;
          }
          else if (meris_type == TIM_MERSYN_MERIS_TYPE_FR)
          {
            ip.osy.list[iband]->pixelWidth  =
              TIM_FMT_ENVISAT_MERIS_FR_PIXEL_SIZE;
          }
        }
      }
/*-------------------------------------------------------------------------*/
      ip.osy.list[iband]->radioShift       = 0.0;
      ip.osy.list[iband]->radioFactor      = 1.0;
      ip.osy.list[iband]->thematicLut.num  = 0;
      ip.osy.list[iband]->thematicLut.list = NULL;
      ip.osy.list[iband]->channel          = ip.cha.list[iband];
      ip.osy.list[iband]->histo.num        = 0;
      ip.osy.list[iband]->histo.list       = NULL;
    } /* Loop on bands */
#ifdef TIM_DEBUG
    fprintf (TimStdout,"\nSETTING PIXEL SIZE DEFAULT VALUES\n");
    fprintf (TimStdout,"   PIXEL HEIGHT = %f\n",
      ip.osy.list[0]->pixelHeight);
    fprintf (TimStdout,"   PIXEL WIDTH  = %f\n",
      ip.osy.list[0]->pixelWidth);
#endif
  } /* When the input synthesis is not provided ... */
/*-------------------------------------------------------------------------*/
/* NDVI output image                                                       */
/*-------------------------------------------------------------------------*/
  if (selection_algorithm == TIM_MERSYN_SELECTION_ALGORITHM_NDVI_MAXIMUM)
  {
    ip.ovi->dataDepth      = 16;
    ip.ovi->backgroundFlag = TIM_TRUE;
    ip.ovi->background     = 0;
    ndvi_min_pixel         = 1;
    ndvi_max_pixel         = pow((double)2,(double)ip.ovi->dataDepth) - 1;
    ndvi_min_value         = -1.0;
    ndvi_max_value         = 1.0;
    ip.ovi->radioShift     = (ndvi_min_value * ndvi_max_pixel -
                              ndvi_max_value * ndvi_min_pixel) /
                             (ndvi_max_pixel - ndvi_min_pixel);
    ip.ovi->radioFactor    = (ndvi_max_value - ndvi_min_value) /
                             (ndvi_max_pixel - ndvi_min_pixel);
  }
/*************************************************************************/
/* Allocate the geographic and output projections to process conversion  */
/*************************************************************************/
  geographic_projection = NULL;
  synthesis_projection  = NULL;
  if (TimPrjAllocProjection("projection=Geographic ellipsoid=WGS_1984 "
      "datum=WGS_84 unit=radian central_meridian=Greenwich",
      NULL,&geographic_projection) != TIM_OK)
  {
    fprintf(TimStderr,"MERSYN: Cannot allocate geographic projection.\n");
    TimExit (&process,TIM_PRS_ERROR);
  }
  if (TimPrjAllocProjection(ip.osy.list[0]->projection,NULL,
      &synthesis_projection) != TIM_OK)
  {
    fprintf(TimStderr,
      "MERSYN: Cannot allocate output synthesis projection.\n");
    TimExit (&process,TIM_PRS_ERROR);
  }
/*************************************************************************/
/* Build the segment boundary (footprint frontiers) within the projection of */
/* the synthesis image. This enveloppe will enable to compute the intersection*/
/* of each line of the synthesis with the segment speeding up the processing. */
/*************************************************************************/
  boundary.num  = 0;
  boundary.list = NULL;
  if (TimMersynComputeBoundary(
      ip.mer,
      geographic_projection,
      synthesis_projection,
      &boundary) != TIM_OK)
  {
    fprintf(TimStderr,
      "MERSYN: Cannot compute the segment boundary.\n");
    TimExit (&process,TIM_PRS_ERROR);
  }
/*************************************************************************/
/* When the input synthesis is not provided, the output synthesis size and */
/* origin shall be retrieved from the segment boundary list              */
/*************************************************************************/
  if (ip.isy.num == 0)
  {
/*-------------------------------------------------------------------------*/
/*    Get the minimum and maximum coordinates from all the boundary segments */
/*-------------------------------------------------------------------------*/
    x_min = TIM_FLT_MAX;
    x_max = -TIM_FLT_MAX;
    y_min = TIM_FLT_MAX;
    y_max = -TIM_FLT_MAX;
    for (iboundary=0; iboundary<boundary.num; iboundary++)
    {
      for (ipoint=0; ipoint<boundary.list[iboundary].num; ipoint++)
      {
        if (boundary.list[iboundary].list[ipoint].x < x_min)
          x_min = boundary.list[iboundary].list[ipoint].x;
        if (boundary.list[iboundary].list[ipoint].x > x_max)
          x_max = boundary.list[iboundary].list[ipoint].x;
        if (boundary.list[iboundary].list[ipoint].y < y_min)
```

```
            y_min = boundary.list[iboundary].list[ipoint].y;
         if (boundary.list[iboundary].list[ipoint].y > y_max)
            y_max = boundary.list[iboundary].list[ipoint].y;
      }
   }
/*-----------------------------------------------------------------------------*/
/*    Add a small quantity to y_max in order not to cross single upper corner */
/*-----------------------------------------------------------------------------*/
   y_max = y_max + (y_max - y_min) / 10000.;
#ifdef TIM_DEBUG
   fprintf (TimStdout,"\nCOMPUTING IMAGE SIZE AND ORIGIN\n");
   fprintf (TimStdout,"   UL: X = %lf   Y = %lf\n",x_min,y_max);
   fprintf (TimStdout,"   LR: X = %lf   Y = %lf\n",x_max,y_min);
#endif
/*-----------------------------------------------------------------------------*/
/*    Compute the image height                                                 */
/*-----------------------------------------------------------------------------*/
   image_height = y_max - y_min + 1;
#ifdef TIM_DEBUG
   fprintf (TimStdout,"   IMAGE HEIGHT = %lf %s\n",image_height,
      ((strstr(ip.osy.list[0]->projection,"projection=Geographic")==NULL)?
       "metres":"degrees"));
#endif
/*-----------------------------------------------------------------------------*/
/*    Compute the image width                                                  */
/*-----------------------------------------------------------------------------*/
   image_width = x_max - x_min;
#ifdef TIM_DEBUG
   fprintf (TimStdout,"   IMAGE WIDTH  = %lf %s\n",image_width,
      ((strstr(ip.osy.list[0]->projection,"projection=Geographic")==NULL)?
       "metres":"degrees"));
#endif
/*-----------------------------------------------------------------------------*/
/*    Set output synthesys resources                                           */
/*-----------------------------------------------------------------------------*/
   for (iband=0; iband<ip.osy.num; iband++)
   {
      ip.osy.list[iband]->lineNumber   = image_height /
                                         ip.osy.list[iband]->pixelHeight;
      ip.osy.list[iband]->pixelNumber  = image_width /
                                         ip.osy.list[iband]->pixelWidth;
      ip.osy.list[iband]->origin.x     = x_min;
      ip.osy.list[iband]->origin.y     = y_max;
      ip.osy.list[iband]->recordLength =
         TimComputeRecordLength((TimRFile)(ip.osy.list[iband]));
   }
#ifdef TIM_DEBUG
   fprintf (TimStdout,"   LINE  NUMBER = %d\n",ip.osy.list[0]->lineNumber);
   fprintf (TimStdout,"   PIXEL NUMBER = %d\n",ip.osy.list[0]->pixelNumber);
#endif
   } /* When the input synthesis is not provided ... */
/*****************************************************************************/
/* Allocates buffers                                                         */
/*****************************************************************************/
/* List of input and output synthesis buffers                                */
/*---------------------------------------------------------------------------*/
   if ((((ip.isy.num > 0)                                                 &&
        (i_buf_synthesis=(TimBufferInt**)TimAlloc(NULL,band_number,
         sizeof(TimBufferInt*))) == NULL))                                ||
       ((o_buf_synthesis=(TimBufferInt**)TimAlloc(NULL,band_number,
         sizeof(TimBufferInt*))) == NULL))
   {
      fprintf(TimStderr,
         "MERSYN: Cannot allocate input and output synthesis buffers lists.\n");
      TimExit (&process,TIM_PRS_ERROR);
   }
/*-----------------------------------------------------------------------------*/
/* Allocate input and output buffers for each band of the synthesis          */
/*-----------------------------------------------------------------------------*/
   for (iband=0; iband<band_number; iband++)
   {
      if (ip.isy.num > 0)
      {
         if ((i_buf_synthesis[iband]=(TimBufferInt*)TimAllocBufferSequential(
              (TimRFile)(ip.isy.list[iband]),TIM_DATA_TYPE_INT,1,
              TIM_EDGE_INIT_METHOD_NONE)) == NULL)
         {
            fprintf(TimStderr,"MERSYN: Cannot allocate buffers of bands.\n");
            TimExit (&process,TIM_PRS_ERROR);
         }
      }
      if ((o_buf_synthesis[iband]=(TimBufferInt*)TimAllocBufferSequential(
           (TimRFile)(ip.osy.list[iband]),TIM_DATA_TYPE_INT,1,
           TIM_EDGE_INIT_METHOD_NONE)) == NULL)
      {
         fprintf(TimStderr,
            "MERSYN: Cannot allocate output buffer of synthesis.\n");
         TimExit (&process,TIM_PRS_ERROR);
      }
   }
/*-----------------------------------------------------------------------------*/
/* Allocate buffers for each one of the 16 possible channels                 */
/* Buffer lines will be dynamically allocated while reading the line         */
/*-----------------------------------------------------------------------------*/
   for (iband=0; iband<16; iband++)
   {
      if ((i_buf_meris[iband]=(TimMerisBuffer*)TimAlloc(NULL,1,
           sizeof(TimMerisBuffer))) == NULL)
      {
         fprintf(TimStderr,
            "MERSYN: Cannot allocate input buffers for MERIS.\n");
         TimExit (&process,TIM_PRS_ERROR);
      }
      for (i=0; i<TIM_MERIS_BUFFER_MAX_LINE_NUMBER; i++)
      {
         i_buf_meris[iband]->lineIndex[i] = -1;
         i_buf_meris[iband]->line[i]      = NULL;
      }
   }
/*-----------------------------------------------------------------------------*/
/* Input and output NDVI buffers                                             */
/*-----------------------------------------------------------------------------*/
   if (((ip.ivi != NULL)                                                  &&
        ((i_buf_ndvi=(TimBufferFloat*)TimAllocBufferSequential(
          (TimRFile)(ip.ivi),TIM_DATA_TYPE_FLOAT,1,
          TIM_EDGE_INIT_METHOD_NONE)) == NULL))                           ||

       ((ip.ovi != NULL)                                                  &&
        ((o_buf_ndvi=(TimBufferFloat*)TimAllocBufferSequential(
          (TimRFile)(ip.ovi),TIM_DATA_TYPE_FLOAT,1,
          TIM_EDGE_INIT_METHOD_NONE)) == NULL)))
   {
      fprintf(TimStderr,
         "MERSYN: Cannot allocate input and/or output NDVI buffers.\n");
      TimExit (&process,TIM_PRS_ERROR);
   }
/*=============================================================================*/
/* Initialise the prediction/correction histogram                            */
/*=============================================================================*/
#ifdef TIM_PERF
   for (ihisto=0; ihisto<=TIM_MERSYN_PREDICTION_CORRECTION_MAX_ITERATION;
        ihisto++)
   {
      prediction_correction_histo[ihisto] = 0;
   }
#endif
/*****************************************************************************/
/*****************************************************************************/
/* LOOP ON LINES                                                             */
/*****************************************************************************/
   processed_point_number = 0;
   added_point_number     = 0;
/*-----------------------------------------------------------------------------*/
/* Usually the first facet to be processed is the upper-right one            */
/*-----------------------------------------------------------------------------*/
   iline_facet_first      = 0;
   ipixel_facet_first     = TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointNumber - 2;
#ifdef TIM_PERF
   time_line              = current_clock;
   time_line0             = current_clock;
   elapsed_time_facet     = 0;
#endif
   for (iline=0; iline<ip.osy.list[0]->lineNumber; iline++)
   {
      iline_facet  = iline_facet_first;
      ipixel_facet = ipixel_facet_first;
      y_current    = ip.osy.list[0]->origin.y -
                     iline * ip.osy.list[0]->pixelHeight;
      is_first_of_line = TIM_TRUE;
/*****************************************************************************/
/*    Compute the intersection of the segment boundary with the current line  */
/*****************************************************************************/
/*    Loop on points of the segment boundary                                 */
/*===========================================================================*/
      intersection.num  = 0;
      intersection.list = NULL;
      if (TimMersynComputeIntersectionsWithBoundary(
          y_current,
          &boundary,
          &intersection,
          &is_inside,
          &current_intersection) != TIM_OK)
      {
         fprintf(TimStderr,"MERSYN: Cannot Compute the intersection of the "
            "segment boundary with the current line #%d (Y-coord=%lf).\n",
            iline,y_current);
         TimExit (&process,TIM_PRS_ERROR);
      }
/*****************************************************************************/
/*    Read current line from input files                                     */
/*****************************************************************************/
/*    Read input synthesis file(s) and copy values into output buffer(s)     */
/*===========================================================================*/
      for (iband=0; iband<band_number; iband++)
      {
         if (ip.isy.num > 0)
         {
            if (TimReadImageSequential((TimRFile)ip.isy.list[iband],
                (TimBuffer*)i_buf_synthesis[iband]) != TIM_OK)
            {
               fprintf(TimStderr,
                  "MERSYN: Cannot read line %d of band %d from \"%s\".\n",
                  iline,iband,ip.isy.list[iband]->name);
               TimExit (&process,TIM_PRS_ERROR);
            }
            for (ipixel=0; ipixel<ip.isy.list[iband]->pixelNumber; ipixel++)
               TimValueSequential (o_buf_synthesis[iband],iline,ipixel) =
                  TimValueSequential (i_buf_synthesis[iband],iline,ipixel);
         }
         else
         {
            for (ipixel=0; ipixel<ip.osy.list[iband]->pixelNumber; ipixel++)
               TimValueSequential (o_buf_synthesis[iband],iline,ipixel) =
                  ip.osy.list[iband]->background;
         }
      }
/*===========================================================================*/
/*    Read input ndvi (if any) and report values into output buffer          */
/*===========================================================================*/
      if (selection_algorithm == TIM_MERSYN_SELECTION_ALGORITHM_NDVI_MAXIMUM)
      {
         if (ip.ivi != NULL)
         {
            if (TimReadImageSequential((TimRFile)ip.ivi,
                (TimBuffer*)i_buf_ndvi) != TIM_OK)
            {
               fprintf(TimStderr,
                  "MERSYN: Cannot read line %d of input NDVI \"%s\".\n",
                  iline,ip.ivi->name);
               TimExit (&process,TIM_PRS_ERROR);
            }
/*-----------------------------------------------------------------------------*/
/*          Convert pixel values to scientific values                          */
/*-----------------------------------------------------------------------------*/
            for (ipixel=0; ipixel<ip.ivi->pixelNumber; ipixel++)
            {
               if (TimValueSequential(i_buf_ndvi,iline,ipixel) !=
                   ip.ivi->background)
               {
                  TimValueSequential(i_buf_ndvi,iline,ipixel) =
                     ip.ivi->radioShift   +
                     ip.ivi->radioFactor  *
                     TimValueSequential(i_buf_ndvi,iline,ipixel);
               }
            }
/*-----------------------------------------------------------------------------*/
/*          Copy input NDVI values into the output NDVI buffer                 */
/*-----------------------------------------------------------------------------*/
            for (ipixel=0; ipixel<ip.ivi->pixelNumber; ipixel++)
               TimValueSequential (o_buf_ndvi,iline,ipixel) =
                  TimValueSequential (i_buf_ndvi,iline,ipixel);
         }
         else
         {
            for (ipixel=0; ipixel<ip.ovi->pixelNumber; ipixel++)
               TimValueSequential (o_buf_ndvi,iline,ipixel) =
                  ip.ovi->background;
         }
      }
/*****************************************************************************/
/*    LOOP ON PIXELS                                                         */
/*****************************************************************************/
      for (ipixel=0; ipixel<ip.osy.list[0]->pixelNumber; ipixel++)
      {
#ifdef TIM_BENCH
/*-----------------------------------------------------------------------------*/
/*       Look if the current point matches one of the bench points           */
/*-----------------------------------------------------------------------------*/
         bench_index = -1;
         for (ibench=0; (bench_index==-1) &&
```

```
                    (ibench<TimNumber(TIM_MERSYN_BENCH)); ibench++)
        {
           if ((TIM_MERSYN_BENCH[ibench].gcp.x == iline)                    &&
               (TIM_MERSYN_BENCH[ibench].gcp.y == ipixel))
              bench_index = ibench;
        }
#endif
        x_current = ip.osy.list[0]->origin.x +
                    ipixel * ip.osy.list[0]->pixelWidth;
/*=========================================================================*/
/*        Case of point inside the segment                                 */
/*=========================================================================*/
        if (is_inside)
        {
/*=========================================================================*/
/*          Compute the geographical coordinates of the current point and  */
/*          Check that geographic coordinates are valid                    */
/*=========================================================================*/
           prj_coordinate.x = x_current;
           prj_coordinate.y = y_current;
           prj_coordinate.z = 0.0;
           if (TimPrjConvertCoordinate(&prj_coordinate,synthesis_projection,
               &prj_coordinate,geographic_projection) != TIM_OK)
           {
              fprintf(TimStderr,
                 "MERSYN: Cannot convert coordinates of current synthesis "
                 "point (%lf %lf).\n",x_current,y_current);
              TimExit (&process,TIM_PRS_ERROR);
           }
           syn_lon_origin = prj_coordinate.x;
           syn_lat_origin = prj_coordinate.y;
#ifdef TIM_BENCH
           if (bench_index != -1)
           {
              TIM_MERSYN_BENCH[bench_index].easting   = x_current;
              TIM_MERSYN_BENCH[bench_index].northing  = y_current;
              TIM_MERSYN_BENCH[bench_index].longitude = syn_lon_origin;
              TIM_MERSYN_BENCH[bench_index].latitude  = syn_lat_origin;
           }
#endif
           if (prj_coordinate.x != TIM_FLOAT_UNKNOWN)
           {
/*************************************************************************/
/*            PREDICTION / CORRECTION ALGORITHM - INITIALISATION         */
/*************************************************************************/
/*            For the first pixel to be processed per line, start the search */
/*            around the facet matching the first pixel of the previous line */
/*-----------------------------------------------------------------------*/
              if (is_first_of_line)
              {
                 iline_facet  = iline_facet_first;
                 ipixel_facet = ipixel_facet_first;
              }
/*=======================================================================*/
/*            Process the inverse location model                         */
/*            Retrieve the antecedent of the point being processed       */
/*=======================================================================*/
              if ((is_found=TimMersynLocationInverse(
                   syn_lon_origin,
                   syn_lat_origin,
                   &iline_facet,
                   &ipixel_facet,
                   &delta_facet_x,
                   &delta_facet_y,
                   &mer_x_origin,
                   &mer_y_origin)) == TIM_ERROR)
              {
                 fprintf(TimStderr,
                    "MERSYN: Cannot retrieve antecedent of (%lf,%lf).\n",
                    syn_lon_origin,syn_lat_origin);
                 TimExit (&process,TIM_PRS_ERROR);
              }
              mer_x0 = mer_x_origin;
              mer_y0 = mer_y_origin;
#ifdef TIM_BENCH
              if (bench_index != -1)
              {
                 TIM_MERSYN_BENCH[bench_index].merisX0 = mer_x_origin;
                 TIM_MERSYN_BENCH[bench_index].merisY0 = mer_y_origin;
              }
#endif
/*-----------------------------------------------------------------------*/
/*            For the first pixel to be processed per line, keep the facet   */
/*            that has been found to prepare the processing of the first     */
/*            pixel of the next line                                         */
/*-----------------------------------------------------------------------*/
              if ((is_first_of_line) && (is_found))
              {
                 iline_facet_first  = iline_facet;
                 ipixel_facet_first = ipixel_facet;
                 is_first_of_line   = TIM_FALSE;
              }
/*************************************************************************/
/*            PREDICTION / CORRECTION ALGORITHM - LOOP                   */
/*            Scope is here to refine the inverse localisation in order to   */
/*            correct parallax effects                                       */
/*************************************************************************/
              prediction_correction_number = 0;
              mer_x0                       = mer_x_origin;
              mer_y0                       = mer_y_origin;
              done                         = TIM_FALSE;
              while ((process_ortho)                                        &&
                     (is_found)                                             &&
                     (!(done)))
              {
/*-----------------------------------------------------------------------*/
/*                Increment the number of iterations                     */
/*-----------------------------------------------------------------------*/
                 prediction_correction_number =
                    prediction_correction_number + 1;
/*=======================================================================*/
/*                Get altitude at (mer_x0,mer_y0) point                  */
/*=======================================================================*/
/*                Get altitude from DEMs (if any)                        */
/*-----------------------------------------------------------------------*/
                 altitude_found = TIM_FALSE;
                 dem_altitude   = 0.0;
                 if (ip.dem.num > 0)
                 {
                    if ((altitude_found=TimMersynGetAltitudeFromDem(
                         syn_lon_origin,
                         syn_lat_origin,
                         geographic_projection,
                         &(ip.dem),
                         &dem_altitude)) == TIM_ERROR)
                    {
                       fprintf(TimStderr,
                          "MERSYN: Cannot get altitude from DEMs.\n");
                       TimExit (&process,TIM_PRS_ERROR);
                    }
                 }
/*-----------------------------------------------------------------------*/
/*                If not found, interpolate altitude from tie-points grid*/
/*-----------------------------------------------------------------------*/
                 if (!(altitude_found))
                 {
                    dem_altitude =
                       (1 - delta_facet_x) * (1 - delta_facet_y) *
                       TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
                       iline_facet].DemAltitude[ipixel_facet] +
                       (1 - delta_facet_x) *      delta_facet_y  *
                       TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
                       iline_facet].DemAltitude[ipixel_facet+1] +
                            delta_facet_x  * (1 - delta_facet_y) *
                       TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
                       iline_facet+1].DemAltitude[ipixel_facet] +
                            delta_facet_x  *      delta_facet_y  *
                       TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
                       iline_facet+1].DemAltitude[ipixel_facet+1];
                    dem_altitude = dem_altitude *
                       TIM_FMT_ENVISAT_MERIS_SPECIAL.scalingFactorAltitude;
                    altitude_found = TIM_TRUE;
                 }
#ifdef TIM_BENCH
                 if (bench_index != -1)
                 {
                    TIM_MERSYN_BENCH[bench_index].altitude = dem_altitude;
                 }
#endif
/*=======================================================================*/
/*                Process the direct location applying the parallax correction*/
/*=======================================================================*/
                 if (TimMersynLocationDirect(
                     mer_x0,
                     mer_y0,
                     dem_altitude,
                     iline_facet,
                     ipixel_facet,
                     delta_facet_x,
                     delta_facet_y,
                     &syn_lon1,
                     &syn_lat1) != TIM_OK)
                 {
                    fprintf(TimStderr,
                       "MERSYN: Cannot get direct image of (%lf,%lf,%lf).\n",
                       mer_x0,mer_y0,dem_altitude);
                    TimExit (&process,TIM_PRS_ERROR);
                 }
/*=======================================================================*/
/*                Process the inverse location model                     */
/*                Retrieve the antecedent of the point being processed   */
/*=======================================================================*/
                 if ((is_found=TimMersynLocationInverse(
                      syn_lon1,
                      syn_lat1,
                      &iline_facet,
                      &ipixel_facet,
                      &delta_facet_x,
                      &delta_facet_y,
                      &mer_x1,
                      &mer_y1)) == TIM_ERROR)
                 {
                    fprintf(TimStderr,
              "MERSYN: Cannot retrieve antecedent for correction of (%lf,%lf).\n",
                       syn_lon1,syn_lat1);
                    TimExit (&process,TIM_PRS_ERROR);
                 }
/*=======================================================================*/
/*                Process refinement of the MERIS point                  */
/*=======================================================================*/
                 if (!(is_found))
                 {
                    done = TIM_TRUE;
                 }
                 else
                 {
/*-----------------------------------------------------------------------*/
/*                   Process correction                                  */
/*-----------------------------------------------------------------------*/
                    mer_x0 = mer_x0 + (mer_x_origin - mer_x1);
                    mer_y0 = mer_y0 + (mer_y_origin - mer_y1);
/*-----------------------------------------------------------------------*/
/*                   Check that the corrected position stays within segment*/
/*-----------------------------------------------------------------------*/
                    if ((mer_x0 < 0)                                    ||
                        (mer_x0 > header->lineNumber - 1)               ||
                        (mer_y0 < 0)                                    ||
                        (mer_y0 > header->pixelNumber - 1))
                    {
                       done = TIM_TRUE;
                    }
/*-----------------------------------------------------------------------*/
/*                   Compute the facet coordinates matching the new point*/
/*-----------------------------------------------------------------------*/
                    else
                    {
                       iline_facet =
                          (int)(mer_x0 /
                                TIM_FMT_ENVISAT_MERIS_SPECIAL.linesPerTiePt);
                       delta_facet_x = (mer_x0 -
                          iline_facet *
                          TIM_FMT_ENVISAT_MERIS_SPECIAL.linesPerTiePt) /
                          (double)TIM_FMT_ENVISAT_MERIS_SPECIAL.linesPerTiePt;

                       ipixel_facet =
                          (int)(mer_y0 /
                                TIM_FMT_ENVISAT_MERIS_SPECIAL.samplesPerTiePt);
                       delta_facet_y = (mer_y0 -
                          ipixel_facet *
                          TIM_FMT_ENVISAT_MERIS_SPECIAL.samplesPerTiePt) /
                          TIM_FMT_ENVISAT_MERIS_SPECIAL.samplesPerTiePt;
                    }
                 }
/*=======================================================================*/
/*                Check prediction / correction loop termination         */
/*=======================================================================*/
/*                Case of precision reached                              */
/*-----------------------------------------------------------------------*/
                 if ((fabs(mer_x_origin - mer_x1) <= TIM_MERSYN_MAX_ERROR)   &&
                     (fabs(mer_y_origin - mer_y1) <= TIM_MERSYN_MAX_ERROR))
                 {
                    done = TIM_TRUE;
                 }
/*-----------------------------------------------------------------------*/
/*                Case of maximum iteration                              */
/*-----------------------------------------------------------------------*/
                 if (prediction_correction_number >=
                     TIM_MERSYN_PREDICTION_CORRECTION_MAX_ITERATION)
                 {
                    done = TIM_TRUE;
```

```
               }
            } /* PREDICTION / CORRECTION ALGORITHM - LOOP */
#ifdef TIM_PERF
            prediction_correction_histo[prediction_correction_number] =
               prediction_correction_histo[prediction_correction_number] + 1;
#endif
            iline_segment  = TimRound((float)mer_x0);
            ipixel_segment = TimRound((float)mer_y0);
#ifdef TIM_BENCH
            if (bench_index != -1)
            {
               TIM_MERSYN_BENCH[bench_index].merisX = mer_x0;
               TIM_MERSYN_BENCH[bench_index].merisY = mer_y0;
/*---------------------------------------------------------------------*/
/*               Compute the (lon,lat) correction at the new MERIS location  */
/*               (mer_x0,mer_y0) assuming a null elevation             */
/*---------------------------------------------------------------------*/
               if (TimMersynLocationDirect(
                   mer_x0,
                   mer_y0,
                   (double)0.0,
                   iline_facet,
                   ipixel_facet,
                   delta_facet_x,
                   delta_facet_y,
                   &null_elevation_longitude,
                   &null_elevation_latitude) != TIM_OK)
               {
                   fprintf(TimStderr,
                      "MERSYN: Cannot get null elevation direct image of "
                      "(%lf,%lf,0.0).\n",
                      mer_x0,mer_y0);
                   TimExit (&process,TIM_PRS_ERROR);
               }
               TIM_MERSYN_BENCH[bench_index].longitudeCorrection =
                  TIM_MERSYN_BENCH[bench_index].longitude -
                  null_elevation_longitude;
               TIM_MERSYN_BENCH[bench_index].latitudeCorrection  =
                  TIM_MERSYN_BENCH[bench_index].latitude -
                  null_elevation_latitude;
            }
#endif
/*********************************************************************/
/*            Check that facet has been found and that the coordinates found */
/*            are within the segment                                 */
/*********************************************************************/
            if ((is_found)                                          &&
                (iline_segment >= 0)                                &&
                (iline_segment <  header->lineNumber)               &&
                (ipixel_segment >= 0)                               &&
                (ipixel_segment <  header->pixelNumber))
            {
               process_pixel          = TIM_TRUE;
               processed_point_number = processed_point_number + 1;
               iline_in_buffer        = iline_segment %
                                        TIM_MERIS_BUFFER_MAX_LINE_NUMBER;
/*=====================================================================*/
/*               Process MERIS flags getting the measurement from the MDS(16)*/
/*=====================================================================*/
               if (process_MERIS_flag)
               {
/*---------------------------------------------------------------------*/
/*                  Read value of this pixel                           */
/*---------------------------------------------------------------------*/
                  if (i_buf_meris[15]->lineIndex[iline_in_buffer] !=
                      iline_segment)
                  {
                     if (TimMersynReadLine(ip.mer,15,iline_segment,
                         i_buf_meris[15]) != TIM_OK)
                     {
                        fprintf (TimStderr,
                           "MERSYN: Unable to get pixel flag.\n");
                        TimExit (&process,TIM_PRS_ERROR);
                     }
                  }
                  flag_value =
                     i_buf_meris[15]->line[iline_in_buffer][ipixel_segment];
                  flag_value = flag_value >> 8;
/*---------------------------------------------------------------------*/
/*                  Check that pixel is flagged as valid               */
/*---------------------------------------------------------------------*/
                  if (flag_value                                    &
                      (TIM_FMT_ENVISAT_QUALITY_FLAG_GLINT_RISK |
/* stderr
                       TIM_FMT_ENVISAT_QUALITY_FLAG_BRIGHT     |
*/
                       TIM_FMT_ENVISAT_QUALITY_FLAG_INVALID))
                     process_pixel = TIM_FALSE;
               } /* Process MERIS flags */
/*********************************************************************/
/*               Case of "Maximum NDVI" algorithm                    */
/*********************************************************************/
               if ((process_pixel)                                  &&
                   (selection_algorithm ==
                    TIM_MERSYN_SELECTION_ALGORITHM_NDVI_MAXIMUM))
               {
/*---------------------------------------------------------------------*/
/*                  Get the Red value                                  */
/*---------------------------------------------------------------------*/
                  if (i_buf_meris[channel_index_red]->lineIndex[
                      iline_in_buffer] != iline_segment)
                  {
                     if (TimMersynReadLine(ip.mer,channel_index_red,
                         iline_segment,i_buf_meris[channel_index_red]) !=
                         TIM_OK)
                     {
                        fprintf (TimStderr,
                     "MERSYN: Unable to get red pixel (%d,%d) from \"%s\".\n",
                           iline_segment,ipixel_segment,ip.mer->name);
                        TimExit (&process,TIM_PRS_ERROR);
                     }
                  }
                  radiance_new_red = i_buf_meris[channel_index_red]->line[
                     iline_in_buffer][ipixel_segment];
/*---------------------------------------------------------------------*/
/*                  Get the NIR value                                  */
/*---------------------------------------------------------------------*/
                  if (i_buf_meris[channel_index_NIR]->lineIndex[
                      iline_in_buffer] != iline_segment)
                  {
                     if (TimMersynReadLine(ip.mer,channel_index_NIR,
                         iline_segment,i_buf_meris[channel_index_NIR]) !=
                         TIM_OK)
                     {
                        fprintf (TimStderr,
                     "MERSYN: Unable to get NIR pixel (%d,%d) from \"%s\".\n",
                           iline_segment,ipixel_segment,ip.mer->name);
                        TimExit (&process,TIM_PRS_ERROR);
                     }
                  }
                  radiance_new_NIR = i_buf_meris[channel_index_NIR]->line[
                     iline_in_buffer][ipixel_segment];
/*---------------------------------------------------------------------*/
/*                  Compute the new NDVI value                         */
/*---------------------------------------------------------------------*/
                  if (radiance_new_red + radiance_new_NIR == 0)
                     ndvi_new = 0.;
                  else
                     ndvi_new = (float)(radiance_new_NIR - radiance_new_red)/
                                (float)(radiance_new_NIR + radiance_new_red);
/*---------------------------------------------------------------------*/
/*                  Compute the previous NDVI value checking for background  */
/*---------------------------------------------------------------------*/
                  if (ip.ivi == NULL)
                  {
                     is_background = TIM_TRUE;
                  }
                  else if (TimValueSequential(i_buf_ndvi,iline,ipixel) ==
                           ip.ivi->background)
                  {
                     is_background = TIM_TRUE;
                  }
                  else
                  {
                     is_background = TIM_FALSE;
                     ndvi_old = TimValueSequential(i_buf_ndvi,iline,ipixel);
                  }
/*---------------------------------------------------------------------*/
/*                  Case of new NDVI greater than the previous one (if any)  */
/*---------------------------------------------------------------------*/
                  if ((is_background)                                   ||
                      (ndvi_new > ndvi_old))
                  {
/*---------------------------------------------------------------------*/
/*                     Report the new values for all the bands         */
/*---------------------------------------------------------------------*/
                     for (iband=0; iband<band_number; iband++)
                     {

                        channel_index = ip.osy.list[iband]->channel - 1;
/*---------------------------------------------------------------------*/
/*                        When the line is not in buffer, read it      */
/*---------------------------------------------------------------------*/
                        if (i_buf_meris[channel_index]->lineIndex[
                            iline_in_buffer] != iline_segment)
                        {
                           if (TimMersynReadLine(ip.mer,channel_index,
                               iline_segment,i_buf_meris[channel_index]) !=
                               TIM_OK)
                           {
                              fprintf (TimStderr,
                           "MERSYN: Unable to get pixel (%d,%d) from \"%s\".\n",
                                 iline_segment,ipixel_segment,ip.mer->name);
                              TimExit (&process,TIM_PRS_ERROR);
                           }
                        }
/*---------------------------------------------------------------------*/
/*                        Report value                                 */
/*---------------------------------------------------------------------*/
                        TimValueSequential(o_buf_synthesis[iband],iline,
                           ipixel) = i_buf_meris[channel_index]->line[
                           iline_in_buffer][ipixel_segment];
                     } /* Report the new values for all the bands */
/*---------------------------------------------------------------------*/
/*                     Keep this new value of the NDVI                 */
/*---------------------------------------------------------------------*/
                     TimValueSequential(o_buf_ndvi,iline,ipixel) =
                        ndvi_new;
                     added_point_number = added_point_number + 1;
                  } /* Case of new NDVI greater than the previous one */
               } /* Case of "Maximum NDVI" algorithm */
/*********************************************************************/
/*               Case of "Minimum radiance" algorithm                */
/*********************************************************************/
               else if ((process_pixel)                             &&
                        (selection_algorithm ==
                         TIM_MERSYN_SELECTION_ALGORITHM_RADIANCE_MINIMUM))
               {
/*---------------------------------------------------------------------*/
/*                  Loop on radiance of each band                      */
/*---------------------------------------------------------------------*/
                  is_replaced = TIM_FALSE;
                  for (iband=0; iband<band_number; iband++)
                  {
                     channel_index = ip.osy.list[iband]->channel - 1;
/*---------------------------------------------------------------------*/
/*                     When the line is not in buffer, read it         */
/*---------------------------------------------------------------------*/
                     if (i_buf_meris[channel_index]->lineIndex[
                         iline_in_buffer] != iline_segment)
                     {
                        if (TimMersynReadLine(ip.mer,channel_index,
                            iline_segment,i_buf_meris[channel_index]) !=
                            TIM_OK)
                        {
                           fprintf (TimStderr,
                        "MERSYN: Unable to get pixel (%d,%d) from \"%s\".\n",
                              iline_segment,ipixel_segment,ip.mer->name);
                           TimExit (&process,TIM_PRS_ERROR);
                        }
                     }
/*---------------------------------------------------------------------*/
/*                     Chek that the new value is less than the previous one */
/*---------------------------------------------------------------------*/
                     if (ip.isy.num > 0)
                     {
                        radiance_old = TimValueSequential(
                           i_buf_synthesis[iband],iline,ipixel);
                     }
                     else
                     {
                        radiance_old = ip.osy.list[iband]->background;
                     }
                     radiance_new = i_buf_meris[channel_index]->line[
                        iline_in_buffer][ipixel_segment];
                     if ((radiance_old == ip.osy.list[iband]->background)  ||
                         (radiance_new < radiance_old))
                     {
/*---------------------------------------------------------------------*/
/*                        Report value                                 */
/*---------------------------------------------------------------------*/
                        TimValueSequential(o_buf_synthesis[iband],iline,
                           ipixel) = radiance_new;
                        is_replaced = TIM_TRUE;
                     } /* Chek that the new value is less ... */
                  } /* Loop on radiance of each band */
/*---------------------------------------------------------------------*/
/*                  When at least one band has been replaced, increment count*/
/*---------------------------------------------------------------------*/
                  if (is_replaced)
```

```
                    added_point_number = added_point_number + 1;
                } /* Case of "Minimum radiance" algorithm */

             } /* Check that facet has been found */

          } /* Check that geographic coordinates are valid */

       } /* Case of point inside the segment */
/*=========================================================================*/
/*       When the next point is crossing the segment boundary -> invert the   */
/*       "is_inside" flag                                                  */
/*=========================================================================*/
       while ((current_intersection < intersection.num)                  &&
              (intersection.list[current_intersection] <= x_current))
       {
          is_inside           = ! is_inside;
          current_intersection = current_intersection + 1;
       }
    } /* LOOP ON PIXELS */
/***************************************************************************/
/*    Write current line into output files                                 */
/***************************************************************************/
/*    Synthesis image bands                                                */
/*=========================================================================*/
    for (iband=0; iband<band_number; iband++)
    {
       if (TimWriteImageSequential((TimRFile)ip.osy.list[iband],
           (TimBuffer*)o_buf_synthesis[iband]) != TIM_OK)
       {
          fprintf(TimStderr,
             "MERSYN: Cannot write line %d of band %d into \"%s\".\n",
             iline,iband,ip.osy.list[iband]->name);
          TimExit (&process,TIM_PRS_ERROR);
       }
    }
/*=========================================================================*/
/*    NDVI maximum                                                         */
/*=========================================================================*/
    if (selection_algorithm == TIM_MERSYN_SELECTION_ALGORITHM_NDVI_MAXIMUM)
    {
/*-------------------------------------------------------------------------*/
/*       Transform scientific values into pixel values                     */
/*-------------------------------------------------------------------------*/
       for (ipixel=0; ipixel<ip.isy.list[0]->pixelNumber; ipixel++)
       {
          if (TimValueSequential(o_buf_ndvi,iline,ipixel) !=
              ip.ovi->background)
          {
             pixel_value = TimRound(
                (TimValueSequential(o_buf_ndvi,iline,ipixel) -
                 ip.ovi->radioShift) / ip.ovi->radioFactor);
             if (pixel_value < ndvi_min_pixel)
                pixel_value = ndvi_min_pixel;
             if (pixel_value > ndvi_max_pixel)
                pixel_value = ndvi_max_pixel;
             TimValueSequential(o_buf_ndvi,iline,ipixel) = pixel_value;
          }
       }
/*-------------------------------------------------------------------------*/
/*       Write line                                                        */
/*-------------------------------------------------------------------------*/
       if (TimWriteImageSequential((TimRFile)ip.ovi,
           (TimBuffer*)o_buf_ndvi) != TIM_OK)
       {
          fprintf(TimStderr,
             "MERSYN: Cannot write line %d into \"%s\".\n",
             iline,ip.ovi->name);
          TimExit (&process,TIM_PRS_ERROR);
       }
    }
/***************************************************************************/
/*    Release memory                                                       */
/***************************************************************************/
    if (intersection.list)
       TimFree(intersection.list);
    intersection.list = NULL;
    intersection.num  = 0;
/***************************************************************************/
/*    Print out line progress                                              */
/***************************************************************************/
#ifdef TIM_PERF
    current_clock          = clock();
    fprintf(TimStdout,"LINE=%5d  elapsed/line=%8ld  "
       "facet search: -same=%5.1f%%  -neighbour=%5.1f%%  -all=%5.1f%%  \r",
       iline,current_clock-time_line,
       ((facet_search_same+facet_search_neighbour+facet_search_all==0) ? 0.0 :
        facet_search_same * 100.0 /
        (facet_search_same + facet_search_neighbour+facet_search_all)),
       ((facet_search_same+facet_search_neighbour+facet_search_all==0) ? 0.0 :
        facet_search_neighbour * 100.0 /
        (facet_search_same + facet_search_neighbour+facet_search_all)),
       ((facet_search_same+facet_search_neighbour+facet_search_all==0) ? 0.0 :
        facet_search_all * 100.0 /
        (facet_search_same + facet_search_neighbour+facet_search_all))
       );
    time_line              = current_clock;
#else
    fprintf (TimStdout,"LINE=%5d \r",iline);
#endif
  } /* LOOP ON LINES */
/***************************************************************************/
/* Print out statistics                                                    */
/***************************************************************************/
  fprintf (TimStdout,"\n");
/*-------------------------------------------------------------------------*/
/* Retrieving of the facet from geographic coordinates                     */
/*-------------------------------------------------------------------------*/
#ifdef TIM_PERF
  fprintf (TimStdout,"\n");
  current_clock = clock();
  fprintf(TimStderr,
     "Total elapsed time=%.3f seconds  "
     "elapsed time/facet=%.3f seconds (%5.1f%%) \n",
     (float)(current_clock - time_line0) / CLOCKS_PER_SEC,
     (float)elapsed_time_facet / CLOCKS_PER_SEC,
     100.0 * elapsed_time_facet / (current_clock - time_line0)
     );
  fprintf (TimStdout,"\n");
#endif
/*-------------------------------------------------------------------------*/
/* Convergence of the prediction/correction loop                           */
/*-------------------------------------------------------------------------*/
#ifdef TIM_PERF
  fprintf (TimStdout,"\n");
  point_number = 0;
  for (ihisto=0; ihisto<=TIM_MERSYN_PREDICTION_CORRECTION_MAX_ITERATION;
       ihisto++)
  {
     point_number = point_number + prediction_correction_histo[ihisto];
  }
  fprintf (TimStdout,"PREDICTION/CORRECTION LOOP STATISTICS  "
     "(total processed points = %d)\n",
     point_number);
  for (ihisto=0; ihisto<=TIM_MERSYN_PREDICTION_CORRECTION_MAX_ITERATION;
       ihisto++)
  {
     fprintf (TimStdout,". ITERATION [%2d] = %10d  (%4.1f %%)\n",
        ihisto,prediction_correction_histo[ihisto],
        100.0 * prediction_correction_histo[ihisto] / (float)point_number);
  }
#endif
/*-------------------------------------------------------------------------*/
/* Number of processed and validated points                                */
/*-------------------------------------------------------------------------*/
  fprintf (TimStdout,"\n");
  if (process_ortho)
     fprintf (TimStdout,"ORTHORECTIFICATION PROCESSING         = YES \n");
  else
     fprintf (TimStdout,"ORTHORECTIFICATION PROCESSING         = NO \n");
  fprintf (TimStdout,"NUMBER OF PROCESSED POINTS IN SEGMENT = %d  "
     "(%.1f%% of synthesis)\n",
     processed_point_number,
     100.*processed_point_number/
     (ip.osy.list[0]->lineNumber*ip.osy.list[0]->pixelNumber));
  fprintf (TimStdout,"NUMBER OF VALIDATED POINTS IN SEGMENT = %d  "
  "(%.1f%% of segment) (%.1f%% of synthesis)\n",
     added_point_number,
     ((processed_point_number==0)?0:100.*added_point_number/
     processed_point_number),
     100.*added_point_number/
     (ip.osy.list[0]->lineNumber*ip.osy.list[0]->pixelNumber));
/*-------------------------------------------------------------------------*/
/* Benchmak points                                                         */
/*-------------------------------------------------------------------------*/
#ifdef TIM_BENCH
  fprintf (TimStdout,"\nBENCHMAK POINTS\n");
  fprintf (TimStdout,"##  LINE PIXEL    GCP LON.     GCP LAT.  GCP ALT.   "
     "EASTING    NORTHING     LONGITUDE    LATITUDE ALTITUDE  "
     "VIEW.ZENITH VIEW.AZIMUTH   LON.CORR.   LAT.CORR.   "
     "ORIGIN: LINE PIXEL   MERIS: LINE PIXEL\n");
  for (ibench=0; ibench<TimNumber(TIM_MERSYN_BENCH); ibench++)
  {
     fprintf (TimStdout,"%2d  %4.0f %4.0f  %12.6lf %12.6lf  %4.0lf  "
        "%11.1lf %11.1lf  %12.6lf %12.6lf  %4.0lf     "
        "%8.3lf    %8.3lf      %9.6lf   %9.6lf   "
        "%8.3lf %8.3lf  %8.3lf %8.3lf\n",
        ibench,
        TIM_MERSYN_BENCH[ibench].gcp.x,
        TIM_MERSYN_BENCH[ibench].gcp.y,
        TIM_MERSYN_BENCH[ibench].gcp.longitude*180/TIM_PI,
        TIM_MERSYN_BENCH[ibench].gcp.latitude*180/TIM_PI,
        TIM_MERSYN_BENCH[ibench].gcp.altitude,
        TIM_MERSYN_BENCH[ibench].easting,
        TIM_MERSYN_BENCH[ibench].northing,
        TIM_MERSYN_BENCH[ibench].longitude*180/TIM_PI,
        TIM_MERSYN_BENCH[ibench].latitude*180/TIM_PI,
        TIM_MERSYN_BENCH[ibench].altitude,
        TIM_MERSYN_BENCH[ibench].viewingZenith*180/TIM_PI,
        TIM_MERSYN_BENCH[ibench].viewingAzimuth*180/TIM_PI,
        TIM_MERSYN_BENCH[ibench].longitudeCorrection*180/TIM_PI,
        TIM_MERSYN_BENCH[ibench].latitudeCorrection*180/TIM_PI,
        TIM_MERSYN_BENCH[ibench].merisX0,
        TIM_MERSYN_BENCH[ibench].merisY0,
        TIM_MERSYN_BENCH[ibench].merisX,
        TIM_MERSYN_BENCH[ibench].merisY);
  }
#endif
/***************************************************************************/
/* Release projection structures                                           */
/***************************************************************************/
  if (geographic_projection)
     TimPrjFreeProjection (geographic_projection);
  geographic_projection = NULL;
  if (synthesis_projection)
     TimPrjFreeProjection (synthesis_projection);
  synthesis_projection = NULL;
/***************************************************************************/
/* Close files                                                             */
/***************************************************************************/
  buf[0] = '\0';
  sprintf (buf,"ORTHORECTIFICATION: %s  SEGMENT POINTS: PROCESSED=%d ADDED=%d",
     ((process_ortho)?"YES":"NO"),processed_point_number,added_point_number);
  if (TimExitProcessing(&process,TIM_FILE_ALL,buf) != TIM_OK)
     TimExit (&process,TIM_PRS_ERROR);
/***************************************************************************/
/* Return                                                                  */
/***************************************************************************/
  TimExit (&process,TIM_PRS_OK);
} /* end of MERSYN process */



/***************************************************************************/
/* TimMersynIsInFacet retrieves the coordinates of the synthesis point inside */
/* the facet specified in input. These coordinates are relative to the facet  */
/* and are in the range [0,1] each one. These values will be used as weight   */
/* for further interpolation within the facet (parameters p and q in MERIS    */
/* documentation volume 11).                                                  */
/*                                                                            */
/*                     (iline_facet,ipixel_facet)                             */
/*                                ^ O-------------------------+               */
/*                                | |                         |               */
/*              delta_facet_x     | |                         |               */
/*                                | |      (syn_lon,syn_lat)  |               */
/*                                v +--------------X          |               */
/*                                  |              |          |               */
/*                                  |              |          |               */
/*                                  |              |          |               */
/*                                  +--------------+----------+               */
/*                                  <------------->                           */
/*                                    delta_facet_y                           */
/*                                                                            */
/* TimMersynIsInFacet returns TIM_TRUE if the point falls inside the facet,  */
/* TIM_FALSE if not and TIM_ERROR when an error occurres.                     */
/***************************************************************************/
int TimMersynIsInFacet
#ifdef TIM_ANSI_PROTOTYPE
(
int                iline_facet,        /* line index of the UL tie-pt corner */
int                ipixel_facet,       /* pixel index of the UL tie-pt corner*/
double             syn_lon,            /* X-coord. point in synthesis */
double             syn_lat,            /* Y-coord. point in synthesis */
double             *delta_facet_x,     /* line-position to compute */
double             *delta_facet_y      /* pixel-position to compute */
)
#else
(iline_facet,ipixel_facet,syn_lon,syn_lat,
 delta_facet_x,delta_facet_y)
int                iline_facet;
int                ipixel_facet;
```

```
double              syn_lon;
double              syn_lat;
double              *delta_facet_x;
double              *delta_facet_y;
#endif
{
/*****************************************************************************/
/* Local variables                                                           */
/*****************************************************************************/
   TimRDoubleVector facet[5];          /* list of facet vertices */
   int              intersection_number;/* number of intersections found */
   double           intersection_list[2];/*list of intersection X-coordinates */
   double           x_intersection;     /* current X-coord. of intersection */
   int              ivertex;            /* index among facet vertices */
   double           inter_point1_x;     /* intersection point X-coordinate */
   double           inter_point1_y;     /* intersection point X-coordinate */
   double           inter_point2_x;     /* along edge intersect.point X-coord.*/
   double           inter_point2_y;     /* along edge intersect.point Y-coord.*/

/*===========================================================================*/
/* Check that facet is included within the tie-point matrix                  */
/*===========================================================================*/
   if ((iline_facet < 0)                                                    ||
       (iline_facet >= TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.num - 1)  ||
       (ipixel_facet < 0)                                                   ||
       (ipixel_facet >= TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointNumber - 1))
      return (TIM_FALSE);
/*===========================================================================*/
/* Report coordinates found in the tie-point matrix                          */
/*===========================================================================*/
   facet[0].x = TIM_FMT_ENVISAT_MERIS_SPECIAL.
                tiePointLine.list[iline_facet].longitude[ipixel_facet];
   facet[0].y = TIM_FMT_ENVISAT_MERIS_SPECIAL.
                tiePointLine.list[iline_facet].latitude[ipixel_facet];
   facet[1].x = TIM_FMT_ENVISAT_MERIS_SPECIAL.
                tiePointLine.list[iline_facet].longitude[ipixel_facet+1];
   facet[1].y = TIM_FMT_ENVISAT_MERIS_SPECIAL.
                tiePointLine.list[iline_facet].latitude[ipixel_facet+1];
   facet[2].x = TIM_FMT_ENVISAT_MERIS_SPECIAL.
                tiePointLine.list[iline_facet+1].longitude[ipixel_facet+1];
   facet[2].y = TIM_FMT_ENVISAT_MERIS_SPECIAL.
                tiePointLine.list[iline_facet+1].latitude[ipixel_facet+1];
   facet[3].x = TIM_FMT_ENVISAT_MERIS_SPECIAL.
                tiePointLine.list[iline_facet+1].longitude[ipixel_facet];
   facet[3].y = TIM_FMT_ENVISAT_MERIS_SPECIAL.
                tiePointLine.list[iline_facet+1].latitude[ipixel_facet];
/*===========================================================================*/
/* Correct the possible overlap of the facet on the -180 / +180 frontier    */
/*===========================================================================*/
/*---------------------------------------------------------------------------*/
/* Case of at least one point located on the +180 degrees East frontier      */
/*---------------------------------------------------------------------------*/
   if( (facet[0].x > TIM_MERSYN_HOT_AREA)                                   ||
       (facet[1].x > TIM_MERSYN_HOT_AREA)                                   ||
       (facet[2].x > TIM_MERSYN_HOT_AREA)                                   ||
       (facet[3].x > TIM_MERSYN_HOT_AREA) )
   {
       if( (facet[0].x<0)                                                   ||
           (facet[1].x<0)                                                   ||
           (facet[2].x<0)                                                   ||
           (facet[3].x<0) )
       {
          if (syn_lon<0)
          {
             if (facet[0].x>0)
                facet[0].x-=2*TIM_PI;
             if (facet[1].x>0)
                facet[1].x-=2*TIM_PI;
             if (facet[2].x>0)
                facet[2].x-=2*TIM_PI;
             if (facet[3].x>0)
                facet[3].x-=2*TIM_PI;
          }
          else
          {
             if (facet[0].x<0)
                facet[0].x+=2*TIM_PI;
             if (facet[1].x<0)
                facet[1].x+=2*TIM_PI;
             if (facet[2].x<0)
                facet[2].x+=2*TIM_PI;
             if (facet[3].x<0)
                facet[3].x+=2*TIM_PI;
          }
       }
   }
/*---------------------------------------------------------------------------*/
/* Case of at least one point located on the -180 degrees West frontier      */
/*---------------------------------------------------------------------------*/
   else
   if( (facet[0].x < -TIM_MERSYN_HOT_AREA)                                  ||
       (facet[1].x < -TIM_MERSYN_HOT_AREA)                                  ||
       (facet[2].x < -TIM_MERSYN_HOT_AREA)                                  ||
       (facet[3].x < -TIM_MERSYN_HOT_AREA) )
   {
       if( (facet[0].x>0)                                                   ||
           (facet[1].x>0)                                                   ||
           (facet[2].x>0)                                                   ||
           (facet[3].x>0) )
       {
          if (syn_lon<0)
          {
             if (facet[0].x>0)
                facet[0].x-=2*TIM_PI;
             if (facet[1].x>0)
                facet[1].x-=2*TIM_PI;
             if (facet[2].x>0)
                facet[2].x-=2*TIM_PI;
             if (facet[3].x>0)
                facet[3].x-=2*TIM_PI;
          }
          else
          {
             if (facet[0].x<0)
                facet[0].x+=2*TIM_PI;
             if (facet[1].x<0)
                facet[1].x+=2*TIM_PI;
             if (facet[2].x<0)
                facet[2].x+=2*TIM_PI;
             if (facet[3].x<0)
                facet[3].x+=2*TIM_PI;
          }
       }
   }
/*---------------------------------------------------------------------------*/
/* Close the polygon duplicating the first point append                      */
/*---------------------------------------------------------------------------*/
   facet[4].x = facet[0].x;
   facet[4].y = facet[0].y;
/*===========================================================================*/
/* Compute the intersection of the horizontal passing through the synthesis  */
/* point with each one of the four segments                                  */
/*===========================================================================*/
   intersection_number = 0;
   for (ivertex=0; (intersection_number < 2) && (ivertex<4); ivertex++)
   {
      if (!(TimDEqual(facet[ivertex].y,facet[ivertex+1].y,(double)1e-8)))
      {
         x_intersection = facet[ivertex].x                          +
                          (facet[ivertex+1].x    - facet[ivertex].x) *
                          (syn_lat - facet[ivertex].y) /
                          (facet[ivertex+1].y    - facet[ivertex].y);
/*---------------------------------------------------------------------------*/
/*       Check that intersection point is within the vertex segment          */
/*---------------------------------------------------------------------------*/
         if (facet[ivertex].x <= facet[ivertex+1].x)
         {
            if ((facet[ivertex].x <= x_intersection)                        &&
                (x_intersection   <  facet[ivertex+1].x))
            {
               intersection_list[intersection_number] = x_intersection;
               intersection_number = intersection_number + 1;
            }
         }
         else
         {
            if ((facet[ivertex].x >= x_intersection)                        &&
                (x_intersection   >  facet[ivertex+1].x))
            {
               intersection_list[intersection_number] = x_intersection;
               intersection_number = intersection_number + 1;
            }
         }
      }
   }
/*===========================================================================*/
/* Check that the synthesis point is between the two intersections           */
/*===========================================================================*/
   if (intersection_number != 2)
      return (TIM_FALSE);
   if (intersection_list[0] <= intersection_list[1])
   {
      if ((syn_lon < intersection_list[0])                      ||
          (syn_lon > intersection_list[1]))
         return (TIM_FALSE);
   }
   else
   {
      if ((syn_lon > intersection_list[0])                      ||
          (syn_lon < intersection_list[1]))
         return (TIM_FALSE);
   }
/*****************************************************************************/
/* Compute the delta_facet_x delta_facet_y weights                           */
/*                                                                           */
/* The solution solves the system:                                           */
/*                                                                           */
/* syn_lon = (1 - delta_facet_x).(1 - delta_facet_y).facet[0].x */
/*                       +      delta_facet_x .(1 - delta_facet_y).facet[3].x */
/*                       + (1 - delta_facet_x).     delta_facet_y .facet[1].x */
/*                       +      delta_facet_x .     delta_facet_y .facet[2].x */
/*                                                                           */
/* syn_lat = (1 - delta_facet_x).(1 - delta_facet_y).facet[0].y */
/*                       +      delta_facet_x .(1 - delta_facet_y).facet[3].y */
/*                       + (1 - delta_facet_x).     delta_facet_y .facet[1].y */
/*                       +      delta_facet_x .     delta_facet_y .facet[2].y */
/*****************************************************************************/
/* Compute delta_facet_y                                                     */
/*****************************************************************************/
/* Check if (S0S3) and (S1S2) are not parrallel                              */
/*---------------------------------------------------------------------------*/
   if (fabs((facet[0].x - facet[3].x) * (facet[1].y - facet[2].y) -
            (facet[0].y - facet[3].y) * (facet[1].x - facet[2].x)) >
       TIM_MERSYN_PARALLELISM_THRESHOLD)
   {
/*---------------------------------------------------------------------------*/
/*    Compute intersection I1 between (S0S3) and (S1S2)                      */
/*---------------------------------------------------------------------------*/
      TimMersynComputeLineIntersection (facet[0].x,facet[0].y,facet[3].x,
         facet[3].y,facet[1].x,facet[1].y,facet[2].x,facet[2].y,
         &inter_point1_x,&inter_point1_y);
/*---------------------------------------------------------------------------*/
/*    Compute intersection I2 between (S0S1) and (I1S)                       */
/*---------------------------------------------------------------------------*/
      TimMersynComputeLineIntersection (facet[0].x,facet[0].y,facet[1].x,
         facet[1].y,syn_lon,syn_lat,
         inter_point1_x,inter_point1_y,
         &inter_point2_x,&inter_point2_y);
   }
/*---------------------------------------------------------------------------*/
/* Case of parallele edges                                                   */
/*---------------------------------------------------------------------------*/
   else
   {
/*---------------------------------------------------------------------------*/
/*    Compute intersection I1 between (S0S1) and the parallel to (S0S3)      */
/*    passing through S                                                      */
/*---------------------------------------------------------------------------*/
      TimMersynComputeLineIntersection (facet[0].x,facet[0].y,facet[1].x,
         facet[1].y,syn_lon,syn_lat,
         syn_lon + facet[0].x - facet[3].x,
         syn_lat + facet[0].y - facet[3].y,
         &inter_point2_x,&inter_point2_y);
   }
/*---------------------------------------------------------------------------*/
/* Compute delta_facet_y                                                     */
/*---------------------------------------------------------------------------*/
   *delta_facet_y =
      TimEuclidianDistance(inter_point2_x,inter_point2_y,facet[0].x,facet[0].y)/
      TimEuclidianDistance(facet[0].x,facet[0].y,facet[1].x,facet[1].y);
/*****************************************************************************/
/* Compute delta_facet_x                                                     */
/*****************************************************************************/
/* Check if (S0S1) and (S2S3) are not parrallel                              */
/*---------------------------------------------------------------------------*/
   if (fabs((facet[0].x - facet[1].x) * (facet[2].y - facet[3].y) -
            ((facet[0].y - facet[1].y) * (facet[2].x - facet[3].x))) >
       TIM_MERSYN_PARALLELISM_THRESHOLD)
   {
/*---------------------------------------------------------------------------*/
/*    Compute intersection I1 between (S0S1) and (S2S3)                      */
/*---------------------------------------------------------------------------*/
      TimMersynComputeLineIntersection (facet[0].x,facet[0].y,facet[1].x,
         facet[1].y,facet[2].x,facet[2].y,facet[3].x,facet[3].y,
         &inter_point1_x,&inter_point1_y);
/*---------------------------------------------------------------------------*/
/*    Compute intersection I2 between (S0S3) and (I1S)                       */
/*---------------------------------------------------------------------------*/
      TimMersynComputeLineIntersection(facet[0].x,facet[0].y,facet[3].x,
         facet[3].y,syn_lon,syn_lat,
         inter_point1_x,inter_point1_y,&inter_point2_x,&inter_point2_y);
```

```
   }
/*-------------------------------------------------------------------------*/
/* Case of parallele edges                                                 */
/*-------------------------------------------------------------------------*/
   else
   {
/*-------------------------------------------------------------------------*/
/*    Compute intersection I1 between (S0S3) and the parallel to (S0S1)    */
/*    passing through S                                                    */
/*-------------------------------------------------------------------------*/
      TimMersynComputeLineIntersection(facet[0].x,facet[0].y,facet[3].x,
         facet[3].y,
         syn_lon,syn_lat,
         syn_lon+facet[0].x-facet[1].x,
         syn_lat+facet[0].y-facet[1].y,
         &inter_point2_x,&inter_point2_y);
   }
/*-------------------------------------------------------------------------*/
/* Compute delta_facet_x                                                   */
/*-------------------------------------------------------------------------*/
   *delta_facet_x =
      TimEuclidianDistance(inter_point2_x,inter_point2_y,facet[0].x,facet[0].y)/
      TimEuclidianDistance(facet[0].x,facet[0].y,facet[3].x,facet[3].y);
/*************************************************************************/
/* Return "Found" status                                                 */
/*************************************************************************/
   return (TIM_TRUE);
} /* TimMersynIsInFacet */



/*************************************************************************/
/* TimMersynComputeLineIntersection computes the location of the concurrence  */
/* point of two lines defined each one by a couple of points given as    */
/* parameters.                                                           */
/*************************************************************************/
int TimMersynComputeLineIntersection
#ifdef TIM_ANSI_PROTOTYPE
(
double              point0_x,           /* X-coordinate of point 1 of D1 */
double              point0_y,           /* Y-coordinate of point 1 of D1 */
double              point1_x,           /* X-coordinate of point 2 of D1 */
double              point1_y,           /* Y-coordinate of point 2 of D1 */
double              point2_x,           /* X-coordinate of point 1 of D2 */
double              point2_y,           /* Y-coordinate of point 1 of D2 */
double              point3_x,           /* X-coordinate of point 2 of D2 */
double              point3_y,           /* Y-coordinate of point 2 of D2 */
double              *point_x_p,         /* X-coordinate of concurrence point */
double              *point_y_p          /* Y-coordinate of concurrence point */
)
#else
(point0_x,point0_y,point1_x,point1_y,point2_x,point2_y,point3_x,point3_y,
 point_x_p,point_y_p)
double              point0_x;           /* X-coordinate of point 1 of D1 */
double              point0_y;           /* Y-coordinate of point 1 of D1 */
double              point1_x;           /* X-coordinate of point 2 of D1 */
double              point1_y;           /* Y-coordinate of point 2 of D1 */
double              point2_x;           /* X-coordinate of point 1 of D2 */
double              point2_y;           /* Y-coordinate of point 1 of D2 */
double              point3_x;           /* X-coordinate of point 2 of D2 */
double              point3_y;           /* Y-coordinate of point 2 of D2 */
double              *point_x_p;         /* X-coordinate of concurrence point */
double              *point_y_p;         /* Y-coordinate of concurrence point */
#endif
{
/*************************************************************************/
/* Local variables                                                       */
/*************************************************************************/
   double           scalar_product;     /* (point0,point1) . (point2,point3) */

/*=======================================================================*/
/* Compute and check scalar product                                      */
/*=======================================================================*/
   scalar_product = (point0_x - point1_x) * (point3_y - point2_y) -
                    (point2_x - point3_x) * (point1_y - point0_y);
   if (TimDEqual(scalar_product,(double)0.0,(double)1.e-12))
   {
      fprintf (TimStderr,"TimMersynComputeLineIntersection: "
        "Lines [(%lf,%lf) (%lf,%lf)] and [(%lf,%lf) (%lf,%lf)] are parallel.\n",
       point0_x,point0_y,point1_x,point1_y,point2_x,point2_y,point3_x,point3_y);
      return (TIM_ERROR);
   }
/*=======================================================================*/
/* Compute coordinates                                                   */
/*=======================================================================*/
   *point_y_p = ((point0_x * (point1_y - point0_y) +
                  point0_y * (point0_x - point1_x)) * (point3_y - point2_y) -
                ((point2_x * (point3_y - point2_y) +
                  point2_y * (point2_x - point3_x)) * (point1_y - point0_y))) /
                scalar_product;

   *point_x_p = ((point0_x * (point1_y - point0_y) +
                  point0_y * (point0_x - point1_x)) -
                 *point_y_p * (point0_x - point1_x)) / (point1_y - point0_y);
/*=======================================================================*/
/* Return "Ok" status                                                    */
/*=======================================================================*/
   return (TIM_OK);
} /* TimMersynComputeLineIntersection */



/*************************************************************************/
/* TimMersynReadLine reads the specified line and store the result into the   */
/* provided buffer.                                                      */
/*************************************************************************/
int TimMersynReadLine
#ifdef TIM_ANSI_PROTOTYPE
(
TimRForeign         mer,                /* MERIS segment in input */
int                 iband,              /* band to be processed (0:first) */
int                 iline,              /* line to be read */
TimMerisBuffer      *meris_buffer_p     /* MERIS buffer to initialize */
)
#else
(mer,iband,iline,meris_buffer_p)
TimRForeign         mer;
int                 iband;
int                 iline;
TimMerisBuffer      *meris_buffer_p;
#endif
{
/*************************************************************************/
/* Local variables                                                       */
/*************************************************************************/
   int              record_length;      /* input record length=2*pixelNumber */
   off64_t          offset;             /* start of data in target file */
   int              iline_in_buffer;    /* nb of the line within MERIS buffer */
   int              ipixel;             /* loop among pixels in line buffer */
   int              status;             /* status of I/O functions */
```

```
/*=======================================================================*/
/* Check parameters                                                      */
/*=======================================================================*/
#ifdef TIM_DEBUG
   if ((meris_buffer_p == NULL)                                          ||
       (meris_buffer_p->lineIndex[iline % TIM_MERIS_BUFFER_MAX_LINE_NUMBER] ==
        iline))
   {
      fprintf (TimStderr,"TimMersynReadLine: Wrong parameters.\n");
      return (TIM_ERROR);
   }
#endif
   iline_in_buffer = iline % TIM_MERIS_BUFFER_MAX_LINE_NUMBER;
/*=======================================================================*/
/* Seek on target file                                                   */
/*=======================================================================*/
   record_length = TIM_FMT_ENVISAT_MERIS_SPECIAL.DSD[3+iband].DsrSize;
   offset        = TIM_FMT_ENVISAT_MERIS_SPECIAL.DSD[3+iband].DsOffset +
                   iline * record_length + 13;
   if (TimFSeek(mer->timFp,(off64_t)offset,TIM_SEEK_SET) != 0)
   {
      fprintf (TimStderr,"TimMersynReadLine: Cannot seek on file \"%s\", "
         "offset=%.0lf.\n",mer->name,(double)offset);
      return (TIM_ERROR);
   }
/*=======================================================================*/
/* When the buffer is not allocated, allocate it                         */
/*=======================================================================*/
   if (meris_buffer_p->line[iline_in_buffer] == NULL)
   {
      if ((meris_buffer_p->line[iline_in_buffer]=(unsigned short*)TimAlloc(
           NULL,record_length,sizeof(unsigned short))) == NULL)
      {
         fprintf (TimStderr,
            "TimMersynReadLine: Cannot allocate %d unsigned shorts.\n",
            record_length);
         return (TIM_ERROR);
      }
   }
/*=======================================================================*/
/* Read record                                                           */
/*=======================================================================*/
   if ((status=TimFRead(meris_buffer_p->line[iline_in_buffer],1,
        record_length-13,mer->timFp))!= record_length-13)
   {
      fprintf (TimStderr,
         "TimMersynReadLine: Cannot read %d characters from file \"%s\".\n",
          record_length,mer->name);
      perror ("TimMersynReadLine");
      return (TIM_ERROR);
   }
   meris_buffer_p->lineIndex[iline_in_buffer] = iline;
/*=======================================================================*/
/* Case of LSBF (Least Significant Byte First)                           */
/*=======================================================================*/
   if (!(TimShortIsMSBF))
   {
      for (ipixel=0; ipixel<TIM_FMT_ENVISAT_MERIS_SPECIAL.lineLength; ipixel++)
      {
         TimShortByteSwap(meris_buffer_p->line[iline_in_buffer][ipixel]);
      }
   }
/*=======================================================================*/
/* Return "Ok" status                                                    */
/*=======================================================================*/
   return (TIM_OK);
} /* TimMersynReadLine */



/*************************************************************************/
/* TimMersynComputeBoundary builts the segment boundary (footprint frinters) */
/* within the projection of the synthesis image.                         */
/* This enveloppe will enable to compute the intersection of each line of the */
/* synthesis with the segment speeding up the processing.                */
/*                                                                       */
/* Tie-point information is to be found within the global variable       */
/* TimFmtEnvisatMerisSpecial  TIM_FMT_ENVISAT_MERIS_SPECIAL which is defined  */
/* in TimFmtEnvisatMeris.h inclusion file.                               */
/*************************************************************************/
int TimMersynComputeBoundary
#ifdef TIM_ANSI_PROTOTYPE
(
TimRForeign         meris_file,         /* MERIS segment in input */
TimPrjProjection    *geographic_projection, /* projection of tie-points */
TimPrjProjection    *synthesis_projection,  /* output image projection */
TimRPolyVectorList  *boundary_p         /* enveloppe to be computed */
)
#else
(meris_file,geographic_projection,synthesis_projection,boundary_p)
TimRForeign         meris_file;
TimPrjProjection    *geographic_projection;
TimPrjProjection    *synthesis_projection;
TimRPolyVectorList  *boundary_p;
#endif
{
/*************************************************************************/
/* Local variables                                                       */
/*************************************************************************/
   TimPrjCoordinate prj_coordinate;     /* convertion prj coordinate struct */
   int              iboundary;          /* index among boundaries in list */
   int              ipoint;             /* index among points in list */
   int              iline;              /* index among lines in image */
   int              ipixel;             /* index among columns in image */
   int              iforward;           /* index within a list in forward wise*/
   int              ibackward;          /* index within a list in backward wis*/
   int              i;                  /* convenient index */

/*************************************************************************/
/* Allocate a first segment enveloppe. A-priori only one is required     */
/*************************************************************************/
   boundary_p->num = 1;
   if ((boundary_p->list=(TimRVectorList*)TimAlloc(NULL,boundary_p->num,
        sizeof(TimRVectorList))) == NULL)
   {
      fprintf(TimStderr,"TimMersynComputeBoundary: "
        "Cannot allocate the list of boundary enveloppes.\n");
      return (TIM_ERROR);
   }
/*=======================================================================*/
/* Allocate a list of points to represent the segment enveloppe          */
/* This polygone is closed (first point is copied at the end)            */
/*=======================================================================*/
   boundary_p->list[0].num  =
      2 * (TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.num - 1 +
           TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointNumber - 1) + 1;
   boundary_p->list[0].list = NULL;
   if ((boundary_p->list[0].list=(TimRVector*)TimAlloc(NULL,
        boundary_p->list[0].num,sizeof(TimRVector))) == NULL)
```

```
   {
      fprintf(TimStderr,"TimMersynComputeBoundary: "
         "Cannot allocate the boundary list of %d vertices.\n",
         boundary_p->list[0].num);
      return (TIM_ERROR);
   }
/*====================================================================*/
/* Initialise the list of vertices of the segment boundary from the points    */
/* around the tie-point matrix                                        */
/*====================================================================*/
   for (ipoint=0; ipoint<boundary_p->list[0].num; ipoint++)
   {
/*====================================================================*/
/*    Determine the edge being processed                              */
/*====================================================================*/
/*    Case of left vertical border (top to bottom)                    */
/*--------------------------------------------------------------------*/
      if (ipoint < TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.num - 1)
      {
         iline  = ipoint;
         ipixel = 0;
      }
/*--------------------------------------------------------------------*/
/*    Case of last line (left to right)                               */
/*--------------------------------------------------------------------*/
      else if (ipoint < TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.num - 1 +
                        TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointNumber - 1)
      {
         iline  = TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.num - 1;
         ipixel = ipoint - TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.num + 1;
      }
/*--------------------------------------------------------------------*/
/*    Case of last vertical border (bottom to top)                    */
/*--------------------------------------------------------------------*/
      else if (ipoint < TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.num - 1 +
                        TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointNumber - 1 +
                        TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.num - 1)
      {
         iline  = TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.num - 1    -
                  (ipoint                                               -
                   TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.num + 1 -
                   TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointNumber + 1);
         ipixel = TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointNumber - 1;
      }
/*--------------------------------------------------------------------*/
/*    Case of first line (right to left)                              */
/*--------------------------------------------------------------------*/
      else
      {
         iline  = 0;
         ipixel = TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointNumber - 1-
                  (ipoint                                          -
                   TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.num + 1 -
                   TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointNumber+ 1-
                   TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.num + 1 );
      }
/*--------------------------------------------------------------------*/
/*    Report tie-point values                                         */
/*--------------------------------------------------------------------*/
      boundary_p->list[0].list[ipoint].x =
       TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[iline].longitude[ipixel];
      boundary_p->list[0].list[ipoint].y =
       TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[iline].latitude[ipixel];
   } /* Initialise the list of vertices of the segment boundary */
/*====================================================================*/
/* Detect the breaks -180 / +180 degrees                              */
/*====================================================================*/
   iboundary = 0;
   iforward  = 0;
   ibackward = boundary_p->list[iboundary].num - 1;
   while (iforward < ibackward)
   {
/*--------------------------------------------------------------------*/
/*    Case of -180 / +180 break                                       */
/*--------------------------------------------------------------------*/
      if (fabs((double)(boundary_p->list[iboundary].list[iforward+1].x -
                        boundary_p->list[iboundary].list[iforward  ].x)) >
          TIM_MERSYN_BREAK_DETECTION)
      {
/*--------------------------------------------------------------------*/
/*       Look for the other break from the end of the current list    */
/*--------------------------------------------------------------------*/
         while (fabs((double)(boundary_p->list[iboundary].list[ibackward  ].x -
                              boundary_p->list[iboundary].list[ibackward-1].x))
                 <= TIM_MERSYN_BREAK_DETECTION)
         {
            ibackward = ibackward - 1;
            if (ibackward <= iforward)
            {
               fprintf(TimStderr,"TimMersynComputeBoundary: "
                  "Error in break detection algorithm.\n");
               return (TIM_ERROR);
            }
         } /* Look for the other break from the end of the current list */
/*--------------------------------------------------------------------*/
/*       Split the current boundary reporting the [iforward,ibackward]*/
/*       vertices within an extra list                                */
/*--------------------------------------------------------------------*/
         boundary_p->num = boundary_p->num + 1;
         if ((boundary_p->list=(TimRVectorList*)TimAlloc(boundary_p->list,
              boundary_p->num,sizeof(TimRVectorList))) == NULL)
         {
            fprintf(TimStderr,"TimMersynComputeBoundary: "
               "Cannot allocate one more boundary enveloppe.\n");
            return (TIM_ERROR);
         }
         boundary_p->list[boundary_p->num-1].num = ibackward - iforward + 1 + 1;
         boundary_p->list[boundary_p->num-1].list = NULL;
         if ((boundary_p->list[boundary_p->num-1].list=(TimRVector*)TimAlloc(
              NULL,boundary_p->list[boundary_p->num-1].num,sizeof(TimRVector)))
             == NULL)
         {
            fprintf(TimStderr,"TimMersynComputeBoundary: "
               "Cannot allocate the extra list of vertices.\n");
            return (TIM_ERROR);
         }
         memcpy (&(boundary_p->list[boundary_p->num-1].list[0]),
                 &(boundary_p->list[iboundary].list[iforward]),
                 (boundary_p->list[boundary_p->num-1].num-1) *
                  sizeof(TimRVector));
/*--------------------------------------------------------------------*/
/*       Point iforward and ibackward are present in both lists       */
/*       within the new list add (or substrat) 180 degrees            */
/*--------------------------------------------------------------------*/
         if (boundary_p->list[iboundary].list[iforward].x  <
             boundary_p->list[iboundary].list[iforward+1].x)
         {
            boundary_p->list[boundary_p->num-1].list[0].x =
               boundary_p->list[boundary_p->num-1].list[0].x + 2 * TIM_PI;
            boundary_p->list[boundary_p->num-1].list[
               boundary_p->list[boundary_p->num-1].num-2].x =
               boundary_p->list[boundary_p->num-1].list[
                  boundary_p->list[boundary_p->num-1].num-2].x + 2 * TIM_PI;
         }
         else
         {
            boundary_p->list[boundary_p->num-1].list[0].x =
               boundary_p->list[boundary_p->num-1].list[0].x - 2 * TIM_PI;
            boundary_p->list[boundary_p->num-1].list[
               boundary_p->list[boundary_p->num-1].num-2].x =
               boundary_p->list[boundary_p->num-1].list[
                  boundary_p->list[boundary_p->num-1].num-2].x - 2 * TIM_PI;
         }
/*--------------------------------------------------------------------*/
/*       Close the new list duplicating the first point at the end    */
/*--------------------------------------------------------------------*/
         memcpy (&(boundary_p->list[boundary_p->num-1].list[
                    boundary_p->list[boundary_p->num-1].num-1]),
                 &(boundary_p->list[boundary_p->num-1].list[0]),
                 sizeof(TimRVector));
/*--------------------------------------------------------------------*/
/*       Remove the points within the current list                    */
/*--------------------------------------------------------------------*/
         for (i=ibackward; i<boundary_p->list[iboundary].num; i++)
            memcpy (
               &(boundary_p->list[iboundary].list[i-(ibackward-iforward)+1]),
               &(boundary_p->list[iboundary].list[i]),
               sizeof(TimRVector));

         boundary_p->list[iboundary].num  = boundary_p->list[iboundary].num -
                                            (ibackward - iforward - 1);
         boundary_p->list[iboundary].list = (TimRVector*)TimAlloc(
            boundary_p->list[iboundary].list,boundary_p->list[iboundary].num,
            sizeof(TimRVector));
/*--------------------------------------------------------------------*/
/*       Current boundary becomes the new one                         */
/*--------------------------------------------------------------------*/
         iboundary = iboundary + 1;
         iforward  = 0;
         ibackward = boundary_p->list[iboundary].num - 1;
      } /* Case of -180 / +180 break */
      iforward = iforward + 1;
   } /* Detect the breaks -180 / +180 degrees */
/*====================================================================*/
/* Print out coordinates within a ".geo" file                         */
/*====================================================================*/
#ifdef TIM_DEBUG
{
   TimFILE         *tim_fp;        /* "MER_...N1.geo" file pointer */

   sprintf(buf,"%s.geo",meris_file->name);
   if ((tim_fp=TimFOpen(buf,TIM_WRITE_ONLY)) == NULL)
   {
      fprintf(TimStderr,"TimMersynComputeBoundary: Cannot open \"%s\" file.\n",
         buf);
      return (TIM_ERROR);
   }
   for (iboundary=0; iboundary<boundary_p->num; iboundary++)
   {
      for (ipoint=0; ipoint<boundary_p->list[iboundary].num; ipoint++)
      {
         TimFPrintf (tim_fp,"%12.8f %12.8f\n",
            boundary_p->list[iboundary].list[ipoint].x * 180.0 / TIM_PI,
            boundary_p->list[iboundary].list[ipoint].y * 180.0 / TIM_PI);
      }
      TimFPrintf (tim_fp,"END\n");
   }
   TimFClose (tim_fp);
   tim_fp = NULL;
}
#endif
/*====================================================================*/
/* Loop on boundary points conversion: Geographic -> Destination projection   */
/*====================================================================*/
   for (iboundary=0; iboundary<boundary_p->num; iboundary++)
   {
      for (ipoint=0; ipoint<boundary_p->list[iboundary].num; ipoint++)
      {
         prj_coordinate.x = boundary_p->list[iboundary].list[ipoint].x;
         prj_coordinate.y = boundary_p->list[iboundary].list[ipoint].y;
         prj_coordinate.z = 0.0;
         if (TimPrjConvertCoordinate(&prj_coordinate,geographic_projection,
             &prj_coordinate,synthesis_projection) != TIM_OK)
         {
            fprintf(TimStderr,"TimMersynComputeBoundary: "
               "Cannot convert coordinates of boundary point #%d (%lf,%lf).\n",
               ipoint,
               boundary_p->list[iboundary].list[ipoint].x,
               boundary_p->list[iboundary].list[ipoint].y);
            return (TIM_ERROR);
         }
         boundary_p->list[iboundary].list[ipoint].x = prj_coordinate.x;
         boundary_p->list[iboundary].list[ipoint].y = prj_coordinate.y;
      } /* Loop of points of the current boundary */
   } /* Loop on boundary points conversion */
/*====================================================================*/
/* Return "Ok" status                                                 */
/*====================================================================*/
   return (TIM_OK);
} /* TimMersynComputeBoundary */



/*****************************************************************************/
/* TimMersynComputeIntersectionsWithBoundary computes the intersection between*/
/* the current horizontal line and the boundary of the segment.              */
/*                                                                           */
/* Intersection points will be stored in the "intersection" list passed in   */
/* parameter.                                                                */
/* TimMersynComputeIntersectionsWithBoundary will also determine if the first */
/* point to be processed on the line (0.0,y_current) is inside a polygon or  */
/* not.                                                                      */
/*****************************************************************************/
int TimMersynComputeIntersectionsWithBoundary
#ifdef TIM_ANSI_PROTOTYPE
(
double              y_current,          /* curr.point Y-coord in output proje.*/
TimRPolyVectorList  *boundary_p,        /* enveloppe of MERIS segment */
TimRFloatList       *intersection_p,    /* list of intersect. segment % line */
int                 *is_inside_p,       /* "point is inside segment" flag */
int                 *current_intersection_p  /* intersection to be reached */
)
#else
(y_current,boundary_p,intersection_p,is_inside_p,current_intersection_p)
double              y_current;
TimRPolyVectorList  *boundary_p;
TimRFloatList       *intersection_p;
int                 *is_inside_p;
int                 *current_intersection_p;
#endif
```

```
{
/*****************************************************************************/
/* Local variables                                                           */
/*****************************************************************************/
   int              iboundary;          /* index among boundaries in list */
   int              ipoint;             /* index among points in list */
   int              i;                  /* convenient index */
   int              j;                  /* convenient index */
   double           tmp_double;         /* temporary variable: double */

/*****************************************************************************/
/* Loop on points of the segment boundary                                    */
/*****************************************************************************/
   intersection_p->num  = 0;
   intersection_p->list = NULL;
   for (iboundary=0; iboundary<boundary_p->num; iboundary++)
   {
      for (ipoint=1; ipoint<boundary_p->list[iboundary].num; ipoint++)
      {
/*---------------------------------------------------------------------------*/
/*    Check that arc is crossing the line                                    */
/*---------------------------------------------------------------------------*/
         if (((boundary_p->list[iboundary].list[ipoint-1].y >= y_current)   &&
              (boundary_p->list[iboundary].list[ipoint].y   <  y_current))    ||
             ((boundary_p->list[iboundary].list[ipoint-1].y <= y_current)   &&
              (boundary_p->list[iboundary].list[ipoint].y   >  y_current)))
         {
            tmp_double = boundary_p->list[iboundary].list[ipoint-1].x +
               (boundary_p->list[iboundary].list[ipoint].x       -
                boundary_p->list[iboundary].list[ipoint-1].x)     *
               (y_current                                        -
                boundary_p->list[iboundary].list[ipoint-1].y)     /
               (boundary_p->list[iboundary].list[ipoint].y -
                boundary_p->list[iboundary].list[ipoint-1].y);
            intersection_p->num = intersection_p->num + 1;
            if ((intersection_p->list=(TimRFloat*)TimAlloc(intersection_p->list,
                 intersection_p->num,sizeof(TimRFloat))) == NULL)
            {
               fprintf(TimStderr,"TimMersynComputeIntersectionsWithBoundary: "
                  "Cannot allocate %d intersection points.\n",
                  intersection_p->num);
               return (TIM_ERROR);
            }
            intersection_p->list[intersection_p->num-1] = tmp_double;
         } /* Check that arc is crossing the line */
      } /* Loop on points of the segment boundary */
   } /* Loop on boundaries */
/*===========================================================================*/
/* Sort the list of intersection points                                      */
/*===========================================================================*/
   for (i=0; i<intersection_p->num-1; i++)
   {
      for (j=i+1; j<intersection_p->num; j++)
      {
         if (intersection_p->list[j] < intersection_p->list[i])
         {
            tmp_double             = intersection_p->list[i];
            intersection_p->list[i] = intersection_p->list[j];
            intersection_p->list[j] = tmp_double;
         }
      }
   }
/*===========================================================================*/
/* Determine if the first point of the line is inside the segment            */
/*===========================================================================*/
   *is_inside_p           = TIM_FALSE;
   *current_intersection_p = 0;
   while ((*current_intersection_p < intersection_p->num)                    &&
          (intersection_p->list[*current_intersection_p] <= 0.0))
   {
      *is_inside_p           = ! (*is_inside_p);
      *current_intersection_p = *current_intersection_p + 1;
   }
/*===========================================================================*/
/* Return "Ok" status                                                        */
/*===========================================================================*/
   return (TIM_OK);
} /* TimMersynComputeIntersectionsWithBoundary */



/*****************************************************************************/
/* TimMersynLocationInverse computes the coordinates (mer_x,mer_y) of the    */
/* pixel in MERIS segment in input from the passed (longitude,latitude)      */
/* geographic coordinates.                                                   */
/*                                                                           */
/* TimMersynLocationInverse tries to retrieve the tie-point facet that       */
/* includes the passed (longitude,latitude).                                 */
/*                                                                           */
/* TimMersynLocationInverse returns:                                         */
/* . TIM_TRUE when the point has been retrieve in the MERTIS segment,        */
/* . TIM_FALSE when no one facet includes this point, and                    */
/* . TIM_ERROR when an error has occurred.                                   */
/*                                                                           */
/* When the point has been retrieved, TimMersynLocationInverse initialises the*/
/* coordinates (mer_x,mer_y) of the point within the segment, the indices of */
/* the facet (iline_facet,ipixel_facet) and also the coordinates of the point */
/* in the facet (delta_facet_x,delta_facet_y).                               */
/*                                                                           */
/* For optimisation reasons, the facet indices (iline_facet,ipixel_facet) are */
/* given by the caller in order to enable a research around the previous     */
/* facet and according to this heuristic:                                    */
/* 1. check for the same facet,                                              */
/* 2. check the 4-connex neighbours,                                         */
/* 3. check the 8-connex neighbours,                                         */
/* 4. check all the facets.                                                  */
/*****************************************************************************/
int TimMersynLocationInverse
#ifdef TIM_ANSI_PROTOTYPE
(
double              longitude,          /* X-coord of point to be processed */
double              latitude,           /* Y-coord of point to be processed */
int                 *iline_facet_p,     /* number of the tie-point frame */
int                 *ipixel_facet_p,    /* number of the tie-point column */
double              *delta_facet_x_p,   /* X-coordinates within facet [0,1] */
double              *delta_facet_y_p,   /* Y-coordinates within facet [0,1] */
double              *mer_x_p,           /* X-coordinates in MERIS segment */
double              *mer_y_p            /* Y-coordinates in MERIS segment */
)
#else
(longitude,latitude,iline_facet_p,ipixel_facet_p,delta_facet_x_p,
 delta_facet_y_p,mer_x_p,mer_y_p)
double              longitude;
double              latitude;
int                 *iline_facet_p;
int                 *ipixel_facet_p;
double              *delta_facet_x_p;
double              *delta_facet_y_p;
double              *mer_x_p;
double              *mer_y_p;
#endif
{
/*****************************************************************************/
/* Local variables                                                           */
/*****************************************************************************/
   int              ifacet;             /* index among tie-point lines */
   int              jfacet;             /* index among tie-point pixels */
   int              is_found;           /* "facet has been found" flag */

/*****************************************************************************/
/* Test the current facet                                                    */
/*****************************************************************************/
#ifdef TIM_PERF
   current_clock = clock();
   time_facet    = current_clock;
#endif
   is_found = TIM_FALSE;
   if ((is_found=TimMersynIsInFacet(*iline_facet_p,*ipixel_facet_p,
        longitude,latitude,delta_facet_x_p,delta_facet_y_p)) != TIM_FALSE)
   {
      if (is_found == TIM_ERROR)
      {
         fprintf(TimStderr,
            "TimMersynLocationInverse: Cannot check for the current facet.\n");
         return (TIM_ERROR);
      }
#ifdef TIM_PERF
      facet_search_same = facet_search_same + 1;
#endif
   }
/*---------------------------------------------------------------------------*/
/* Test the 4-connex neighbour facets                                        */
/*---------------------------------------------------------------------------*/
   else if ((is_found=TimMersynIsInFacet(*iline_facet_p,*ipixel_facet_p-1,
             longitude,latitude,delta_facet_x_p,delta_facet_y_p)) == TIM_TRUE)
   {
      *ipixel_facet_p = *ipixel_facet_p - 1;
#ifdef TIM_PERF
      facet_search_neighbour = facet_search_neighbour + 1;
#endif
   }
   else if ((is_found=TimMersynIsInFacet(*iline_facet_p-1,*ipixel_facet_p,
             longitude,latitude,delta_facet_x_p,delta_facet_y_p)) == TIM_TRUE)
   {
      *iline_facet_p = *iline_facet_p - 1;
#ifdef TIM_PERF
      facet_search_neighbour = facet_search_neighbour + 1;
#endif
   }
   else if ((is_found=TimMersynIsInFacet(*iline_facet_p+1,*ipixel_facet_p,
             longitude,latitude,delta_facet_x_p,delta_facet_y_p)) == TIM_TRUE)
   {
      *iline_facet_p = *iline_facet_p + 1;
#ifdef TIM_PERF
      facet_search_neighbour = facet_search_neighbour + 1;
#endif
   }
   else if ((is_found=TimMersynIsInFacet(*iline_facet_p,*ipixel_facet_p+1,
             longitude,latitude,delta_facet_x_p,delta_facet_y_p)) == TIM_TRUE)
   {
      *ipixel_facet_p = *ipixel_facet_p + 1;
#ifdef TIM_PERF
      facet_search_neighbour = facet_search_neighbour + 1;
#endif
   }
/*---------------------------------------------------------------------------*/
/* Test the 8-connex neighbour facets                                        */
/*---------------------------------------------------------------------------*/
   else if ((is_found=TimMersynIsInFacet(*iline_facet_p-1,*ipixel_facet_p-1,
             longitude,latitude,delta_facet_x_p,delta_facet_y_p)) == TIM_TRUE)
   {
      *iline_facet_p  = *iline_facet_p  - 1;
      *ipixel_facet_p = *ipixel_facet_p - 1;
#ifdef TIM_PERF
      facet_search_neighbour = facet_search_neighbour + 1;
#endif
   }
   else if ((is_found=TimMersynIsInFacet(*iline_facet_p-1,*ipixel_facet_p+1,
             longitude,latitude,delta_facet_x_p,delta_facet_y_p)) == TIM_TRUE)
   {
      *iline_facet_p  = *iline_facet_p  - 1;
      *ipixel_facet_p = *ipixel_facet_p + 1;
#ifdef TIM_PERF
      facet_search_neighbour = facet_search_neighbour + 1;
#endif
   }
   else if ((is_found=TimMersynIsInFacet(*iline_facet_p+1,*ipixel_facet_p-1,
             longitude,latitude,delta_facet_x_p,delta_facet_y_p)) == TIM_TRUE)
   {
      *iline_facet_p  = *iline_facet_p  + 1;
      *ipixel_facet_p = *ipixel_facet_p - 1;
#ifdef TIM_PERF
      facet_search_neighbour = facet_search_neighbour + 1;
#endif
   }
   else if ((is_found=TimMersynIsInFacet(*iline_facet_p+1,*ipixel_facet_p+1,
             longitude,latitude,delta_facet_x_p,delta_facet_y_p)) == TIM_TRUE)
   {
      *iline_facet_p  = *iline_facet_p  + 1;
      *ipixel_facet_p = *ipixel_facet_p + 1;
#ifdef TIM_PERF
      facet_search_neighbour = facet_search_neighbour + 1;
#endif
   }
/*---------------------------------------------------------------------------*/
/* Case of failure -> Test all the facets                                    */
/*---------------------------------------------------------------------------*/
   else
   {
      is_found = TIM_FALSE;
      for (ifacet=0; (!is_found) && (ifacet <
           TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.num-1);
           ifacet++)
      {
         for (jfacet=0; (!is_found) && (jfacet <
              TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointNumber-1);
              jfacet++)
         {
            if ((is_found=TimMersynIsInFacet(ifacet,jfacet,
                 longitude,latitude,delta_facet_x_p,delta_facet_y_p))==TIM_TRUE)
            {
               *iline_facet_p  = ifacet;
               *ipixel_facet_p = jfacet;
            }
            else if (is_found == TIM_ERROR)
            {
               fprintf(TimStderr,
                  "TimMersynLocationInverse: Cannot check for facet (%d,%d).\n",
                  ifacet,jfacet);
               return (TIM_ERROR);
```

```
         }
      }
   }
#ifdef TIM_PERF
   facet_search_all = facet_search_all + 1;
#endif
  } /* Case of failure -> Test all the facets */
#ifdef TIM_PERF
  current_clock      = clock();
  elapsed_time_facet = elapsed_time_facet +
                       (current_clock - time_facet);
#endif
/*=============================================================================*/
/* Compute coordinates of the source point within the segment                  */
/*=============================================================================*/
  if (is_found)
  {
     *mer_x_p = (*iline_facet_p  + *delta_facet_x_p) *
                TIM_FMT_ENVISAT_MERIS_SPECIAL.linesPerTiePt;
     *mer_y_p = (*ipixel_facet_p + *delta_facet_y_p) *
                TIM_FMT_ENVISAT_MERIS_SPECIAL.samplesPerTiePt;
  }
/*=============================================================================*/
/* Return "IS_FOUND" status                                                    */
/*=============================================================================*/
  return (is_found);
} /* TimMersynLocationInverse */



/*****************************************************************************/
/* TimMersynGetAltitudeFromDem read the altitude value from the list of DEMs. */
/*                                                                           */
/* Altitude is always read from the first DEM first. If the location of the  */
/* point of interest is out of this DEM or if the value found is background, */
/* altitude is read from the second in the list, and so on.,.                */
/*                                                                           */
/* TimMersynGetAltitudeFromDem performs a bi-linear interpolation from the 4 */
/* nearest neighbours of the point of interest.                              */
/*                                                                           */
/* TimMersynGetAltitudeFromDem return:                                       */
/* 1. TIM_TRUE if an altitude has been found in one DEM,                     */
/* 2. TIM_FALSE if no altitude in whatever DEM has been found and            */
/* 3. TIM_ERROR if an error occurred.                                        */
/*****************************************************************************/
int TimMersynGetAltitudeFromDem
#ifdef TIM_ANSI_PROTOTYPE
(
double              longitude,          /* X-coord of point to be processed */
double              latitude,           /* Y-coord of point to be processed */
TimPrjProjection    *geographic_projection, /* projection structure */
TimRDEMList         *dem_p,             /* DEMs in order they have to be read */
double              *altitude_p         /* altitude to be initialized */
)
#else
(longitude,latitude,geographic_projection,dem_p,altitude_p)
double              longitude;
double              latitude;
TimPrjProjection    *geographic_projection;
TimRDEMList         *dem_p;
double              *altitude_p;
#endif
{
/*****************************************************************************/
/* Local variables                                                           */
/*****************************************************************************/
   static TimPrjProjection **dem_projection=NULL; /* DEM proj. struct. list */
   static TimBufferInt     **dem_buf=NULL;        /* DEM direct buffer list */
   TimPrjCoordinate prj_coordinate;     /* convertion prj coordinate struct */
   int              idem;               /* index among DEMs in list */
   double           x_dem;              /* X-coord of interest point in DEM */
   double           y_dem;              /* Y-coord of interest point in DEM */
   int              i0;                 /* truncated value of x_dem */
   int              j0;                 /* truncated value of y_dem */
   TimRDouble       dx;                 /* decimal part of x_dem coordinate */
   TimRDouble       dy;                 /* decimal part of y_dem coordinate */
   TimRDouble       z1;                 /* 1st nearest pixel value in input */
   TimRDouble       z2;                 /* 2nd nearest pixel value in input */
   TimRDouble       z3;                 /* 3rd nearest pixel value in input */
   TimRDouble       z4;                 /* 4th nearest pixel value in input */

/*****************************************************************************/
/* For the first call, initialise the projection structures of each DEM      */
/*****************************************************************************/
   if (dem_projection == NULL)
   {
      if ((dem_projection=(TimPrjProjection**)TimAlloc(NULL,
          dem_p->num,sizeof(TimPrjProjection*))) == NULL)
      {
         fprintf (TimStderr,"TimMersynGetAltitudeFromDem: "
            "Cannot allocate DEM projection list.\n");
         return (TIM_ERROR);
      }
      for (idem=0; idem<dem_p->num; idem++)
      {
         dem_projection[idem] = NULL;
         if (TimPrjAllocProjection(dem_p->list[idem]->projection,NULL,
             &(dem_projection[idem])) != TIM_OK)
         {
            fprintf (TimStderr,"TimMersynGetAltitudeFromDem: "
               "Cannot allocate projection structure for DEM \"%s\".\n",
               dem_p->list[idem]->name);
            return (TIM_ERROR);
         }
      }
   }
/*=============================================================================*/
/* For the first call, initialise the direct buffer of each DEM                */
/*=============================================================================*/
   if (dem_buf == NULL)
   {
      if ((dem_buf=(TimBufferInt**)TimAlloc(NULL,
           dem_p->num,sizeof(TimBufferInt*))) == NULL)
      {
         fprintf (TimStderr,"TimMersynGetAltitudeFromDem: "
            "Cannot allocate DEM buffer list.\n");
         return (TIM_ERROR);
      }
      for (idem=0; idem<dem_p->num; idem++)
      {
         dem_buf[idem] = NULL;
         if ((dem_buf[idem]=(TimBufferInt*)TimAllocBufferDirect(
              TIM_DATA_TYPE_INT)) == NULL)
         {
            fprintf (TimStderr,"TimMersynGetAltitudeFromDem: "
               "Cannot allocate buffer structure for DEM \"%s\".\n",
               dem_p->list[idem]->name);
            return (TIM_ERROR);
         }
      }
   }
/*=============================================================================*/
/* Loop on DEMs                                                                */
/*=============================================================================*/
   for (idem=0; idem<dem_p->num; idem++)
   {
/*-----------------------------------------------------------------------------*/
/*    Convert (longitude,latitude) coordinates to DEM projection               */
/*-----------------------------------------------------------------------------*/
      prj_coordinate.x = longitude;
      prj_coordinate.y = latitude;
      prj_coordinate.z = 0.0;
      if (TimPrjConvertCoordinate(&prj_coordinate,geographic_projection,
          &prj_coordinate,dem_projection[idem]) != TIM_OK)
      {
         fprintf (TimStderr,"TimMersynGetAltitudeFromDem: "
            "Cannot convert geographic coordinates to projection "
            "\"%s\" of DEM \"%s\".\n",dem_p->list[idem]->projection,
            dem_p->list[idem]->name);
         return (TIM_ERROR);
      }
      x_dem = (dem_p->list[idem]->origin.y - prj_coordinate.y) /
              dem_p->list[idem]->pixelHeight;
      y_dem = (prj_coordinate.x - dem_p->list[idem]->origin.x) /
              dem_p->list[idem]->pixelWidth;
      i0    = x_dem;
      j0    = y_dem;
/*-----------------------------------------------------------------------------*/
/*    Check that point is within the DEM                                       */
/*-----------------------------------------------------------------------------*/
      if ((i0 >= 0) && (i0 < dem_p->list[idem]->lineNumber - 1)              &&
          (j0 >= 0) && (j0 < dem_p->list[idem]->pixelNumber - 1))
      {
/*-----------------------------------------------------------------------------*/
/*       Read the four neighbours around the point of interest                 */
/*-----------------------------------------------------------------------------*/
         if (TimReadImageDirect((TimRFile)(dem_p->list[idem]),i0,2,j0,2,1.0,1.0,
             TIM_SAMPLING_METHOD_NEAREST_NEIGHBOUR,NULL,
             (TimBuffer*)dem_buf[idem]) != TIM_OK)
         {
            fprintf (TimStderr,"TimMersynGetAltitudeFromDem: "
               "Cannot extract window ( %d 2 %d 2 ) from DEM \"%s\".\n",
               i0,j0,dem_p->list[idem]->name);
            return (TIM_ERROR);
         }
/*-----------------------------------------------------------------------------*/
/*       Check that pixel is not background                                    */
/*-----------------------------------------------------------------------------*/
         z1 = TimValueDirect(dem_buf[idem],0,0);
         z2 = TimValueDirect(dem_buf[idem],0,1);
         z3 = TimValueDirect(dem_buf[idem],1,0);
         z4 = TimValueDirect(dem_buf[idem],1,1);
         if ((!(dem_p->list[idem]->backgroundFlag))                              ||
             ((z1 != dem_p->list[idem]->background)                              &&
              (z2 != dem_p->list[idem]->background)                              &&
              (z3 != dem_p->list[idem]->background)                              &&
              (z4 != dem_p->list[idem]->background)))
         {
/*-----------------------------------------------------------------------------*/
/*          Convert altitudes to scientific values                             */
/*-----------------------------------------------------------------------------*/
            z1 = z1 * dem_p->list[idem]->radioFactor +
                 dem_p->list[idem]->radioShift;
            z2 = z2 * dem_p->list[idem]->radioFactor +
                 dem_p->list[idem]->radioShift;
            z3 = z3 * dem_p->list[idem]->radioFactor +
                 dem_p->list[idem]->radioShift;
            z4 = z4 * dem_p->list[idem]->radioFactor +
                 dem_p->list[idem]->radioShift;
/*-----------------------------------------------------------------------------*/
/*          Compute bi-linear interpolation                                    */
/*-----------------------------------------------------------------------------*/
            dx = x_dem - i0;
            dy = y_dem - j0;
            *altitude_p = z1  +  dy * (z2 - z1)  +
                                 dx * (z3 + dy * (z4 - z3) -
                                       z1 - dy * (z2 - z1));
/*-----------------------------------------------------------------------------*/
/*          Return "IS_FOUND" status                                           */
/*-----------------------------------------------------------------------------*/
            return (TIM_TRUE);
         } /* Check that pixel is not background */
      } /* Check that point is within the DEM */
   } /* Loop on DEMs */
/*=============================================================================*/
/* Return "NOT_FOUND" status                                                   */
/*=============================================================================*/
  return (TIM_FALSE);
} /* TimMersynGetAltitudeFromDem */



/*****************************************************************************/
/* TimMersynLocationDirect computes the geographic coordinates of the point  */
/* (mer_x0,mer_y0,altitude) given in MERIS segment reference system.         */
/*                                                                           */
/* TimMersynLocationDirect applies the parallax correction interpolating the */
/* viewing angle from the tie-points grid.                                   */
/*****************************************************************************/
int TimMersynLocationDirect
#ifdef TIM_ANSI_PROTOTYPE
(
double              mer_x,              /* X-coordinates in MERIS segment */
double              mer_y,              /* Y-coordinates in MERIS segment */
double              altitude,           /* Z-coordinates in MERIS segment */
int                 iline_facet,        /* number of the tie-point frame */
int                 ipixel_facet,       /* number of the tie-point column */
double              delta_facet_x,      /* X-coordinates within facet [0,1] */
double              delta_facet_y,      /* Y-coordinates within facet [0,1] */
double              *longitude_p,       /* longitude to be computed */
double              *latitude_p         /* latitude to be computed */
)
#else
(mer_x,mer_y,altitude,iline_facet,ipixel_facet,delta_facet_x,delta_facet_y,
 longitude_p,latitude_p)
double              mer_x;
double              mer_y;
double              altitude;
int                 iline_facet;
int                 ipixel_facet;
double              delta_facet_x;
double              delta_facet_y;
double              *longitude_p;
double              *latitude_p;
#endif
{
/*****************************************************************************/
/* Local variables                                                           */
/*****************************************************************************/
   double           longitude;          /* X-coord interpolated from tie-point*/
```

```
   double           latitude;           /* Y-coord interpolated from tie-point*/
   double           viewing_zenith;     /* viewing angle from vertical */
   double           viewing_azimuth;    /* viewing angle from north */
   double           dx;                 /* parallax module */
   double           dlon;               /* longitude correction */
   double           dlat;               /* latitude correction */

/*============================================================================*/
/* Interpolate the geodetic coordinates                                       */
/*============================================================================*/
/* longitude                                                                  */
/*----------------------------------------------------------------------------*/
   longitude =
      (1 - delta_facet_x) * (1 - delta_facet_y) *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet].longitude[ipixel_facet] +
      (1 - delta_facet_x) *      delta_facet_y  *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet].longitude[ipixel_facet+1] +
           delta_facet_x  * (1 - delta_facet_y) *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet+1].longitude[ipixel_facet] +
           delta_facet_x  *      delta_facet_y  *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet+1].longitude[ipixel_facet+1];
/*----------------------------------------------------------------------------*/
/* latitude                                                                   */
/*----------------------------------------------------------------------------*/
   latitude =
      (1 - delta_facet_x) * (1 - delta_facet_y) *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet].latitude[ipixel_facet] +
      (1 - delta_facet_x) *      delta_facet_y  *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet].latitude[ipixel_facet+1] +
           delta_facet_x  * (1 - delta_facet_y) *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet+1].latitude[ipixel_facet] +
           delta_facet_x  *      delta_facet_y  *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet+1].latitude[ipixel_facet+1];
/*============================================================================*/
/* Interpolate the viewing angle                                              */
/*============================================================================*/
/* Viewing zenith                                                             */
/*----------------------------------------------------------------------------*/
   viewing_zenith =
      (1 - delta_facet_x) * (1 - delta_facet_y) *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet].viewingZenith[ipixel_facet] +
      (1 - delta_facet_x) *      delta_facet_y  *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet].viewingZenith[ipixel_facet+1] +
           delta_facet_x  * (1 - delta_facet_y) *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet+1].viewingZenith[ipixel_facet] +
           delta_facet_x  *      delta_facet_y  *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet+1].viewingZenith[ipixel_facet+1];
/*----------------------------------------------------------------------------*/
/* Viewing azimuth                                                            */
/*----------------------------------------------------------------------------*/
   viewing_azimuth =
      (1 - delta_facet_x) * (1 - delta_facet_y) *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet].viewingAzimuth[ipixel_facet] +
      (1 - delta_facet_x) *      delta_facet_y  *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet].viewingAzimuth[ipixel_facet+1] +
           delta_facet_x  * (1 - delta_facet_y) *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet+1].viewingAzimuth[ipixel_facet] +
           delta_facet_x  *      delta_facet_y  *
      TIM_FMT_ENVISAT_MERIS_SPECIAL.tiePointLine.list[
      iline_facet+1].viewingAzimuth[ipixel_facet+1];
#ifdef TIM_BENCH
   if (bench_index != -1)
   {
      TIM_MERSYN_BENCH[bench_index].viewingZenith  = viewing_zenith;
      TIM_MERSYN_BENCH[bench_index].viewingAzimuth = viewing_azimuth;
   }
#endif
/*============================================================================*/
/* Compute the longitude and latitude corrections                             */
/*============================================================================*/
   dx = altitude * tan(viewing_zenith);
   dlat = dx * cos(viewing_azimuth) / TIM_MERSYN_MEAN_EARTH_RADIUS;
   dlon = dx * sin(viewing_azimuth) /
          (TIM_MERSYN_MEAN_EARTH_RADIUS * cos(latitude));
/*============================================================================*/
/* Apply geodetic corrections                                                 */
/*============================================================================*/
   *longitude_p = longitude + dlon;
   *latitude_p  = latitude  + dlat;
/*============================================================================*/
/* Return "Ok" status                                                         */
/*============================================================================*/
   return (TIM_OK);
} /* TimMersynLocationDirect */
```

APPENDIX D – ORTHORECTIFICATION BENCHMARK

This section contains a second version of the log of MERSYN execution (version 02.05) when applied to MERIS FR scene MER_FR__1PNUPA20030921_092341_000000982020_00079_08149_0534.N1 of Italy and using the GETASSE30 DEM above WGS84 ellipsoid directive to produce 25 control points for unitary testing (see e-mail of Norman FOMFERRA, Brockmann Consult on 27/08/2004).

This second version makes suite the comments of Norman FOMFERRA (see R-7) that lead to the modification of equations 6a and 6b.

Print out of the benchmark values is processed compiling the MERSYN program provided in 0 with the “TIM_BENCH” compilation directive.

These values are also stored within a MS EXCEL file “VT-P194-DOC-001-E-01-04.xls” that may be provided upon demand.

D.1 Benchmark points distribution

As shown in fig. 65, benchmark points have been dispatched within the output image to encompass cases with different viewing angles and different elevations.

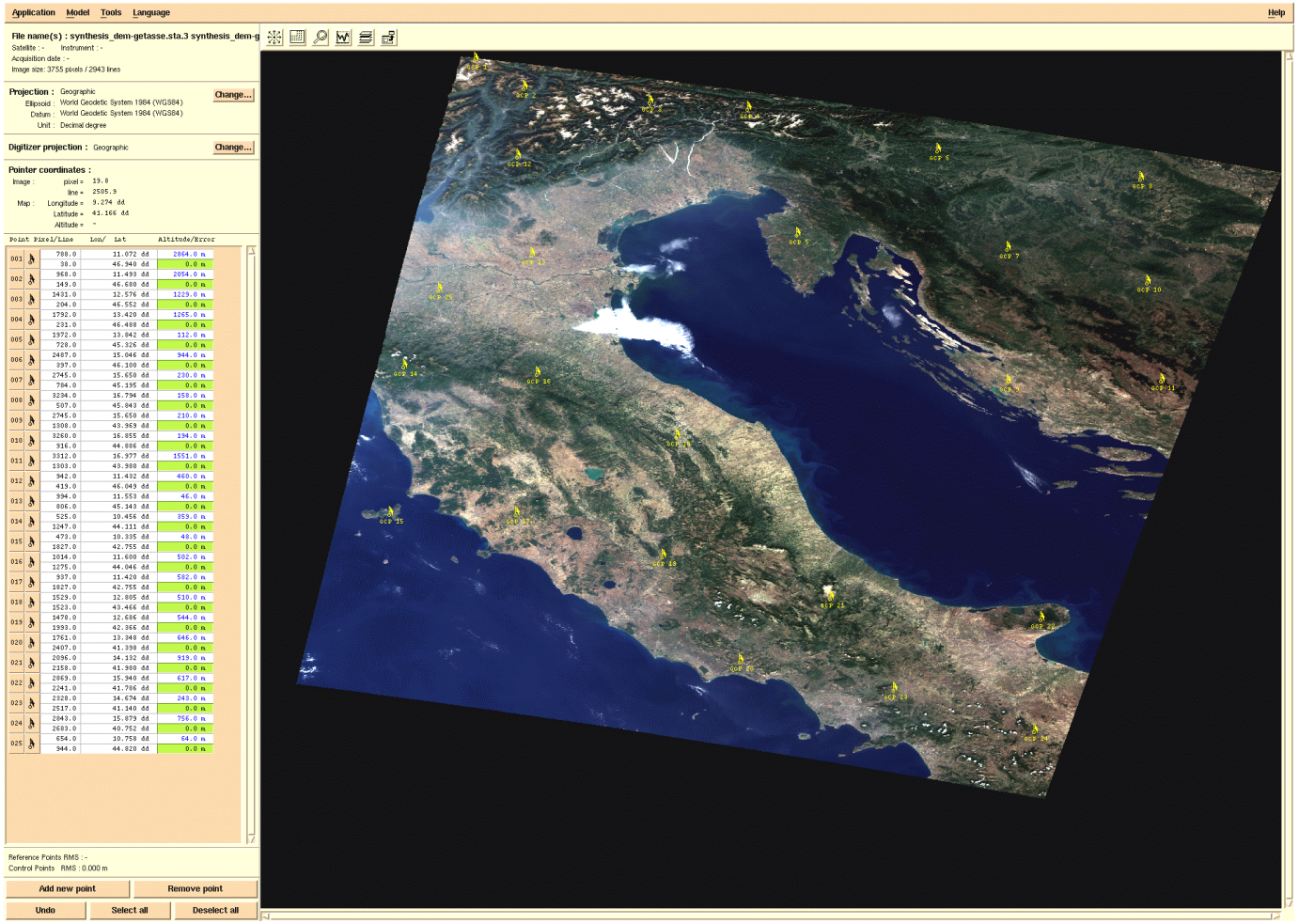


fig. 65 - Distribution of benchmark points.

D.2 Benchmark values

Values are stored within the following columns:

Nb	is a sequential number of the benchmark point.
(line,column)	are the coordinates of the benchmark point within the output (geocoded) image.
(Easting,Northing)	are the coordinates of the benchmark point within the coordinates reference system of the output image: “Plate Carrée – WGS 1984”.
(longitude,latitude)	are the coordinates of the benchmark point within the geodetic reference system (also called “*geographic coordinates*”).
elevation	is the elevation computed by a bi-linear interpolation from the four nearest neighbours at the benchmark point location found within the GETASSE30 DEM. These values are given above WGS 1984 ellipsoid.
view zenith/azimuth	are the two angles interpolated within the tie-point grid at the location of the benchmark point.
longitude/latitude corrections	are the parallax corrections computed taking into account the viewing angles and the point elevation.
MERIS origin (line,column)	are the coordinates of the antecedent of the benchmark point found within the input MERIS scene not taking into account the elevation.
MERIS antecedent (line,column)	are the coordinates of the antecedent of the benchmark point found within the input MERIS scene taking into account the elevation and after having processed the prediction/correction loop.

Nb	line	column	Easting	Northing	longitude	latitude	elevation	view zenith	view azimuth	longitude correction	latitude correction	MERIS origin line	MERIS origin column	MERIS antecedent line	MERIS antecedent column
0	38	788	1230413.2	5216575.0	11.071550	46.939981	2841	38.825	99.946	0.029666	-0.003551	12.362	2098.622	12.375	2107.442
1	149	968	1277213.2	5187715.0	11.492668	46.680292	2036	36.737	100.298	0.019604	-0.002443	90.297	1957.374	90.302	1963.237
2	204	1431	1397593.2	5173415.0	12.575875	46.551617	1424	31.672	101.089	0.011273	-0.001519	85.682	1633.586	85.684	1636.974
3	231	1792	1491453.2	5166395.0	13.420450	46.488449	1436	27.462	101.695	0.009548	-0.001361	65.485	1383.903	65.488	1386.782
4	728	1972	1538253.2	5037175.0	13.841568	45.325697	164	23.385	102.166	0.000884	-0.000134	473.198	1158.107	473.198	1158.380
5	397	2487	1672153.2	5123235.0	15.046432	46.100085	985	18.091	102.895	0.004068	-0.000646	118.239	877.737	118.240	878.979
6	784	2745	1739233.2	5022615.0	15.650033	45.194683	252	12.812	103.431	0.000710	-0.000119	414.628	613.340	414.628	613.561
7	507	3234	1866373.2	5094635.0	16.794069	45.842736	158	7.285	104.152	0.000252	-0.000044	105.166	345.520	105.166	345.598
8	1308	2745	1739233.2	4886375.0	15.650033	43.968763	215	10.309	103.573	0.000474	-0.000082	864.013	490.957	864.013	491.107
9	916	3260	1873133.2	4988295.0	16.854897	44.885863	196	4.804	104.297	0.000203	-0.000037	450.513	227.214	450.513	227.278

Nb	line	column	Easting	Northing	longitude	latitude	elevation	view zenith	view azimuth	longitude correction	latitude correction	MERIS origin line	MERIS origin column	MERIS antecedent line	MERIS antecedent column
10	1303	3312	1886653.2	4887675.0	16.976554	43.980461	1516	2.016	104.478	0.000646	-0.000120	772.409	94.983	772.409	95.189
11	419	942	1270453.2	5117515.0	11.431840	46.048616	522	36.219	100.377	0.004872	-0.000619	329.537	1926.552	329.536	1928.026
12	806	994	1283973.2	5016895.0	11.553496	45.143213	47	34.585	100.619	0.000402	-0.000053	661.958	1820.076	661.958	1820.200
13	1247	525	1162033.2	4902235.0	10.456251	44.111475	459	38.335	100.049	0.004474	-0.000569	1101.176	2070.374	1101.176	2071.773
14	1827	473	1148513.2	4751435.0	10.334594	42.754541	48	37.390	100.208	0.000442	-0.000058	1614.970	2005.610	1614.970	2005.751
15	1275	1014	1289173.2	4894955.0	11.600287	44.045968	559	33.026	100.830	0.004465	-0.000614	1068.898	1718.696	1068.900	1720.097
16	1827	937	1269153.2	4751435.0	11.420142	42.754541	588	32.249	100.918	0.004465	-0.000632	1559.338	1669.616	1559.338	1671.048
17	1523	1529	1423073.2	4830475.0	12.805151	43.465761	473	26.044	101.727	0.002805	-0.000423	1219.772	1304.177	1219.772	1305.069
18	1993	1478	1409813.2	4708275.0	12.685834	42.366177	583	24.994	101.805	0.003241	-0.000500	1633.713	1244.989	1633.714	1246.038
19	2407	1761	1483393.2	4600635.0	13.347924	41.397607	637	19.533	102.362	0.002647	-0.000435	1952.375	951.629	1952.377	952.501
20	2158	2096	1570493.2	4665375.0	14.131671	41.980152	919	15.805	102.796	0.003069	-0.000518	1689.644	760.930	1689.646	761.933
21	2241	2869	1771473.2	4643795.0	15.940137	41.785970	662	3.816	103.997	0.000517	-0.000096	1642.937	180.101	1642.937	180.271
22	2517	2328	1630813.2	4572035.0	14.674444	41.140257	304	10.719	103.244	0.000669	-0.000119	1963.413	510.395	1963.413	510.617
23	2683	2843	1764713.2	4528875.0	15.879308	40.751893	780	1.953	104.063	0.000306	-0.000058	2024.333	92.061	2024.333	92.164
24	944	654	1195573.2	4981015.0	10.758052	44.820356	66	37.769	100.131	0.000634	-0.000080	821.305	2032.557	821.305	2032.753

APPENDIX E – TIE-POINT VALUES

This section contains the values for the first and last (36th for FR or 71st for RR) tie-point of some lines (sampling factor of 8) and the last one (36th for FR or undefined for RR, see example E.15).

E.1 scene01-01/MER_FR__1PNUPA20030921_092217_000000982020_00079_08149_0354.N1

Viewing vectors found in the tie-points ADS enable registering the viewing geometry.

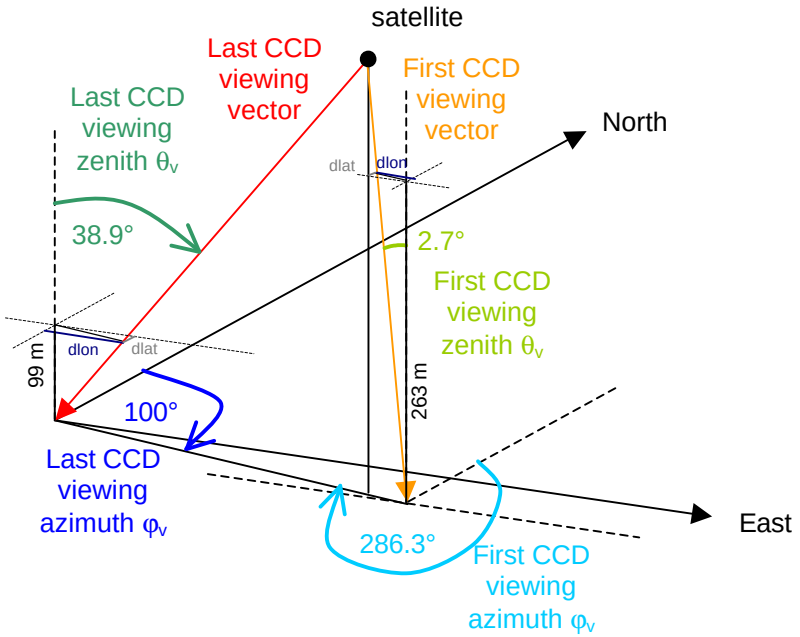


fig. 66 - First and last viewing vectors of the first tie-points line.

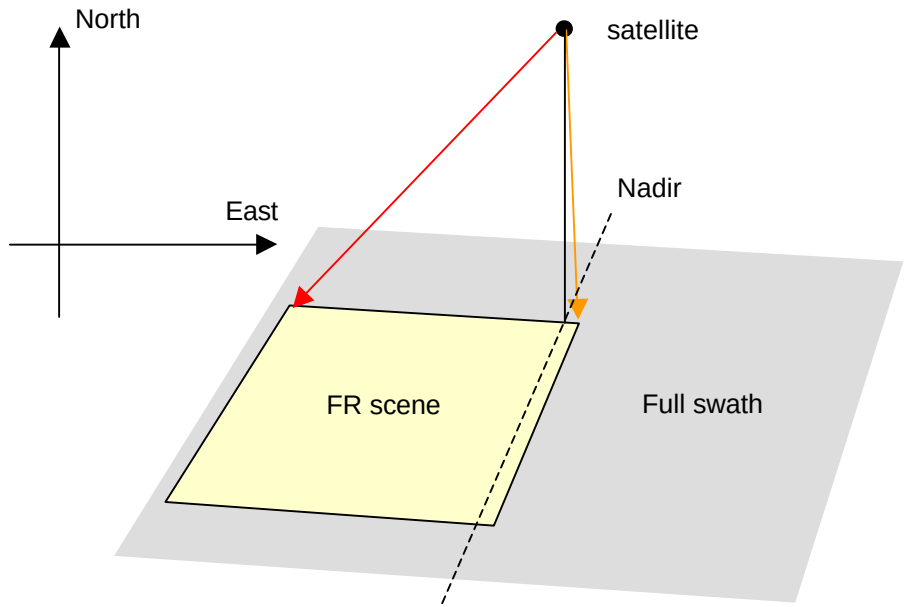


fig. 67 - Location of the FR scene within the full swath in geodetic geometry.

Figure fig. 66 here above illustrates the viewing vector values of the first and last tie-point in the first tie-point line found within the third data set ("Tie-point ADS"). Do not forget that tie-points are ordered from Esat to West. These values let suppose that the FR scene is located on the left side of the full swath as illustrated in fig. 67.

The parallax defect is also provided as "corrected latitude and longitude" According to the MERIS Level 1 Detailed Processing Model (R-3), these corrections are computed using the equation 2.

$$dx = Z_{F,J} \times \tan(\theta_{vF,J})$$
$$dlat_{F,J} = \frac{dx \times \cos(\varphi_{vF,J})}{R_e}$$
$$dlon_{F,J} = \frac{-dx \times \sin(\varphi_{vF,J})}{R_e \times \cos(\phi_{F,J})}$$

Eq. 2

Where:

dx	is the modulus of the parallax error.
$dlat_{F,J}$	is the latitude correction of tie-point (F,J).
$dlon_{F,J}$	is the longitude correction of tie-point (F,J).
$Z_{F,J}$	is the DEM elevation of tie-point (F,J).
$\theta_{vF,J}$	is the zenith of the viewing angle of tie-point (F,J).
$\varphi_{vF,J}$	is the azimuth of the viewing angle of tie-point (F,J).
$\phi_{F,J}$	is the geodetic latitude of tie-point (F,J).
R_e	is the mean Earth radius (R_e = 6 370 997.0 metres)

In our example, see the bold values of the first line of data dumped here below.

FIRST TIE-POINT

$$dx = Z_{F,J} \times \tan(\theta_{vF,J}) = 263 \times \tan(2.7°) = 12.403\ m$$
$$dlat_{F,J} = \frac{dx \times \cos(\varphi_{vF,J})}{R_e} = \frac{12.403 \times \cos(286.3°)}{6370997} = 0.000031306\ °$$
$$dlon_{F,J} = \frac{-dx \times \sin(\varphi_{vF,J})}{R_e \times \cos(\phi_{F,J})} = \frac{-12.403 \times \sin(286.3°)}{6370997 \times \cos(50.742379°)} = 0.000169181\ °$$

LAST TIE-POINT

$$dx = Z_{F,J} \times \tan(\theta_{vF,J}) = 99 \times \tan(38.9°) = 79.883\ m$$
$$dlat_{F,J} = \frac{dx \times \cos(\varphi_{vF,J})}{R_e} = \frac{79.883 \times \cos(100.0°)}{6370997} = -0.000124750\ °$$
$$dlon_{F,J} = \frac{-dx \times \sin(\varphi_{vF,J})}{R_e \times \cos(\phi_{F,J})} = \frac{-79.883 \times \sin(100.0°)}{6370997 \times \cos(51.937422°)} = -0.001147541\ °$$

Sign of the latitude correction (dlat) is not compliant with the sign of longitude correction (dlon). In the case these quantities have to be considered as "CORRECTIONS TO BE ADDED" to the original values (see eq. 6b') and not to be substracted as previously in eq. 6b, expression of dlon computation should be changed as:

$$dlon_{F,J} = \frac{dx \times \sin(\varphi_{vF,J})}{R_e \times \cos(\phi_{F,J})}$$

Eq. 2c'

$$l_{corrected} \leftarrow l_{origin} + dlon_{F,J}$$
$$j_{corrected} \leftarrow j_{origin} + dlat_{F,J}$$

Eq. 6b'

DATA SET #3 - "Tie points ADS":

No.	Date	FIRST (Lat., Lon.)	Alt.	Rough.	Corr. Lat.	Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt.	Rough.	Corr. Lat.	Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	117451337.241756	(**50.742379**, 20.514153)	**263**	27	31 1e-6	169 1e-6	(52.0, 157.9)	(**2.7, 286.3**)	(**51.937422**, 12.372044)	**99**	11	-124 1e-6	-1146 1e-6	(55.4, 148.6)	(**38.9, 100.0**)
9	117451359.768220	(49.435929, 19.924214)	642	91	75 1e-6	403 1e-6	(51.0, 156.9)	(2.7, 286.0)	(50.615694, 12.001195)	433	33	-543 1e-6	-4876 1e-6	(54.4, 147.8)	(38.9, 100.0)
17	117451382.294684	(48.127118, 19.360172)	196	42	22 1e-6	120 1e-6	(49.9, 155.9)	(2.7, 285.7)	(49.293239, 11.641344)	536	20	-671 1e-6	-5875 1e-6	(53.3, 147.0)	(38.9, 99.9)
25	117451404.821148	(46.816133, 18.819646)	159	20	18 1e-6	95 1e-6	(48.8, 154.9)	(2.7, 285.5)	(47.970091, 11.291352)	634	19	-794 1e-6	-6771 1e-6	(52.3, 146.1)	(38.9, 99.9)
33	117451427.347612	(45.503143, 18.300527)	95	10	10 1e-6	55 1e-6	(47.8, 153.8)	(2.7, 285.2)	(46.646280, 10.950213)	1315	442	-1647 1e-6	-13703 1e-6	(51.3, 145.3)	(38.9, 99.9)
36	117451435.795036	(45.010286, 18.110986)	164	49	18 1e-6	95 1e-6	(47.4, 153.4)	(2.7, 285.1)	(46.149686, 10.824383)	1364	358	-1709 1e-6	-14086 1e-6	(51.0, 144.9)	(38.9, 99.9)

E.2 scene01-02/MER_FR__1PNUPA20030921_092220_000000982020_00079_08149_0398.N1

DATA SET #3 - "Tie points ADS":

No.	Date	FIRST (Lat., Lon.)	Alt. Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt. Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	117451340.058163	(49.074601, 27.223613)	300 27	797 1e-6	3090 1e-6 (49.1, 166.0)	(38.9, 291.5)	(50.743669, 19.533811)	271 29	-31 1e-6	-175 1e-6 (52.3, 156.7)	(2.7, 105.6)
9	117451362.584627	(47.800010, 26.479166)	255 22	662 1e-6	2572 1e-6 (48.0, 164.9)	(38.9, 291.0)	(49.433965, 18.970195)	769 175	-86 1e-6	-485 1e-6 (51.2, 155.8)	(2.7, 105.3)
17	117451385.111091	(46.520884, 25.769594)	806 48	2046 1e-6	7965 1e-6 (46.9, 163.8)	(38.9, 290.5)	(48.122164, 18.430685)	199 24	-22 1e-6	-122 1e-6 (50.1, 154.7)	(2.7, 105.1)
25	117451407.637555	(45.237598, 25.091987)	700 76	1738 1e-6	6782 1e-6 (45.8, 162.6)	(38.9, 290.0)	(46.808431, 17.913047)	188 31	-20 1e-6	-113 1e-6 (49.1, 153.7)	(2.7, 104.8)
33	117451430.164019	(43.950485, 24.443732)	78 7	189 1e-6	741 1e-6 (44.7, 161.5)	(38.9, 289.6)	(45.492911, 17.415303)	427 169	-45 1e-6	-251 1e-6 (48.0, 152.7)	(2.7, 104.6)
36	117451438.611443	(43.466894, 24.207718)	137 60	330 1e-6	1293 1e-6 (44.2, 161.0)	(38.9, 289.4)	(44.999157, 17.233411)	213 56	-22 1e-6	-124 1e-6 (47.6, 152.3)	(2.7, 104.5)

E.3 scene01-03/MER_FR__1PNUPA20030921_092327_000000982020_00079_08149_0394.N1

DATA SET #3 - "Tie points ADS":

No.	Date	FIRST (Lat., Lon.)	Alt. Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt. Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	117451407.636284	(46.612105, 18.963019)	93 7	15 1e-6	83 1e-6 (48.6, 155.0)	(4.1, 285.6)	(47.778675, 11.466962)	609 165	-750 1e-6	-6269 1e-6 (52.1, 146.3)	(38.0, 100.1)
9	117451430.162748	(45.299488, 18.441791)	101 2	17 1e-6	88 1e-6 (47.5, 153.9)	(4.1, 285.3)	(46.454782, 11.121505)	1164 222	-1433 1e-6	-11692 1e-6 (51.1, 145.4)	(38.0, 100.1)
17	117451452.689212	(43.984986, 17.940157)	830 189	138 1e-6	711 1e-6 (46.5, 152.9)	(4.1, 285.1)	(45.130236, 10.784263)	20 0	-24 1e-6	-196 1e-6 (50.1, 144.5)	(38.0, 100.1)
25	117451475.215676	(42.668733, 17.456441)	-80 0	0 1e-6	0 1e-6 (45.4, 151.8)	(4.1, 284.9)	(43.805064, 10.454420)	176 81	-217 1e-6	-1688 1e-6 (49.1, 143.5)	(38.0, 100.1)
33	117451497.742140	(41.350854, 16.989140)	-128 0	0 1e-6	0 1e-6 (44.4, 150.6)	(4.1, 284.7)	(42.479293, 10.131249)	-91 0	0 1e-6	0 1e-6 (48.2, 142.6)	(38.0, 100.1)
36	117451506.189564	(40.856253, 16.817854)	362 28	58 1e-6	296 1e-6 (44.0, 150.2)	(4.1, 284.6)	(41.981979, 10.011643)	-567 0	0 1e-6	0 1e-6 (47.8, 142.2)	(38.0, 100.1)

E.4 scene01-04/MER_FR__1PNUPA20030921_092341_000000982020_00079_08149_0534.N1

DATA SET #3 - "Tie points ADS":

No.	Date	FIRST (Lat., Lon.)	Alt. Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt. Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	117451421.714409	(45.909863, 18.014693)	98 3	0 1e-6	0 1e-6 (48.2, 153.6)	(0.0, 182.0)	(47.028196, 10.603496)	1476 382	-1907 1e-6	-16512 1e-6 (51.8, 145.0)	(40.7, 99.6)
9	117451444.240873	(44.594348, 17.519381)	591 159	0 1e-6	0 1e-6 (47.2, 152.5)	(0.0, 181.9)	(45.703992, 10.278711)	850 227	-1101 1e-6	-9284 1e-6 (50.8, 144.1)	(40.7, 99.6)
17	117451466.767337	(43.277165, 17.041762)	417 32	0 1e-6	0 1e-6 (46.1, 151.4)	(0.0, 181.9)	(44.379209, 9.960598)	744 75	-967 1e-6	-7942 1e-6 (49.8, 143.2)	(40.7, 99.7)
25	117451489.293801	(41.958432, 16.580328)	-110 0	0 1e-6	0 1e-6 (45.1, 150.3)	(0.0, 181.9)	(43.053869, 9.648478)	-366 0	0 1e-6	0 1e-6 (48.9, 142.3)	(40.7, 99.7)
33	117451511.820265	(40.638252, 16.133722)	540 177	0 1e-6	0 1e-6 (44.1, 149.2)	(0.0, 181.8)	(41.727992, 9.341743)	116 23	-152 1e-6	-1186 1e-6 (47.9, 141.3)	(40.7, 99.7)
36	117451520.267689	(40.142832, 15.969815)	1053 209	0 1e-6	0 1e-6 (43.7, 148.8)	(0.0, 181.8)	(41.230654, 9.227991)	79 54	-103 1e-6	-801 1e-6 (47.6, 140.9)	(40.7, 99.8)

E.5 scene01-05/MER_FR__1PNUPA20030925_103640_000000982020_00137_08207_0539.N1

DATA SET #3 - "Tie points ADS":

No.	Date	FIRST (Lat., Lon.)	Alt. Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt. Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	117801400.752079	(51.545466, 3.970685)	-2 0	0 1e-6	0 1e-6 (53.8, 161.5)	(12.0, 287.9)	(52.871546, -4.276685)	-12 0	0 1e-6	0 1e-6 (57.0, 152.2)	(32.1, 101.4)
9	117801423.278543	(50.247548, 3.315102)	55 13	31 1e-6	157 1e-6 (52.7, 160.5)	(12.0, 287.5)	(51.551705, -4.707541)	-38 0	0 1e-6	0 1e-6 (55.9, 151.4)	(32.1, 101.3)
17	117801445.805007	(48.946594, 2.690314)	72 27	40 1e-6	201 1e-6 (51.6, 159.5)	(12.0, 287.1)	(50.230855, -5.122747)	75 50	-82 1e-6	-649 1e-6 (54.9, 150.5)	(32.1, 101.2)
25	117801468.331471	(47.642858, 2.093492)	135 6	74 1e-6	368 1e-6 (50.5, 158.5)	(12.0, 286.8)	(48.909059, -5.523896)	-121 0	0 1e-6	0 1e-6 (53.8, 149.7)	(32.1, 101.1)
33	117801490.857935	(46.336567, 1.522125)	311 32	169 1e-6	829 1e-6 (49.4, 157.5)	(12.0, 286.5)	(47.586371, -5.912392)	-140 0	0 1e-6	0 1e-6 (52.8, 148.9)	(32.1, 101.1)
36	117801499.305359	(45.846090, 1.313961)	333 38	180 1e-6	881 1e-6 (49.0, 157.1)	(12.0, 286.4)	(47.090144, -6.055074)	-1669 0	0 1e-6	0 1e-6 (52.4, 148.5)	(32.1, 101.0)

E.6 scene01-06/MER_FR__1PNUPA20030926_100615_000000982020_00151_08221_0538.N1

DATA SET #3 - "Tie points ADS":

No.	Date	FIRST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	117885975.564369	(46.131973, 15.246156)	432	134	1178 1e-6	4509 1e-6 (48.3, 165.1)	(40.7, 290.7)	(47.752857, 7.968492)	907	96	0 1e-6	0 1e-6 (51.5, 156.3)	(0.0, 182.1)
9	117885998.090833	(44.850067, 14.567378)	-90	0	0 1e-6	0 1e-6 (47.2, 164.0)	(40.7, 290.2)	(46.439901, 7.445472)	1549	398	0 1e-6	0 1e-6 (50.4, 155.3)	(0.0, 182.1)
17	117886020.617297	(43.564274, 13.917832)	-55	0	0 1e-6	0 1e-6 (46.0, 162.9)	(40.7, 289.7)	(45.125091, 6.942563)	1579	352	0 1e-6	0 1e-6 (49.3, 154.3)	(0.0, 182.0)
25	117886043.143761	(42.274895, 13.295207)	1103	373	2824 1e-6	10886 1e-6 (44.9, 161.8)	(40.7, 289.3)	(43.808563, 6.458004)	939	98	0 1e-6	0 1e-6 (48.3, 153.3)	(0.0, 182.0)
33	117886065.670225	(40.982206, 12.697413)	-694	0	0 1e-6	0 1e-6 (43.8, 160.7)	(40.7, 288.9)	(42.490438, 5.990218)	-2420	0	0 1e-6	0 1e-6 (47.2, 152.3)	(0.0, 182.0)
36	117886074.117649	(40.496644, 12.479252)	-2251	0	0 1e-6	0 1e-6 (43.4, 160.3)	(40.7, 288.8)	(41.995752, 5.818829)	-2399	0	0 1e-6	0 1e-6 (46.8, 151.9)	(0.0, 181.9)

E.7 scene01-07/MER_FR__1PNUPA20031010_092709_000000982020_00351_08421_0536.N1

DATA SET #3 - "Tie points ADS":

No.	Date	FIRST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	119093229.243887	(43.298630, 18.667244)	1584	288	1035 1e-6	4917 1e-6 (52.0, 159.2)	(14.6, 286.1)	(44.541324, 11.619780)	13	10	-13 1e-6	-93 1e-6 (55.3, 151.3)	(30.0, 101.3)
9	119093251.770351	(41.986534, 18.156891)	-1196	0	0 1e-6	0 1e-6 (50.9, 158.3)	(14.6, 285.9)	(43.216211, 11.255751)	246	80	-248 1e-6	-1723 1e-6 (54.2, 150.6)	(30.0, 101.2)
17	119093274.296815	(40.672518, 17.664457)	143	14	90 1e-6	427 1e-6 (49.8, 157.4)	(14.7, 285.6)	(41.890343, 10.900534)	-1365	0	0 1e-6	0 1e-6 (53.1, 149.8)	(30.0, 101.2)
25	119093296.823279	(39.356718, 17.188507)	-447	0	0 1e-6	0 1e-6 (48.7, 156.5)	(14.7, 285.4)	(40.563758, 10.553316)	-1832	0	0 1e-6	0 1e-6 (52.1, 149.0)	(30.0, 101.2)
33	119093319.349743	(38.039257, 16.727746)	-1717	0	0 1e-6	0 1e-6 (47.6, 155.6)	(14.7, 285.2)	(39.236493, 10.213367)	-1629	0	0 1e-6	0 1e-6 (51.1, 148.1)	(30.1, 101.1)
36	119093327.797167	(37.544803, 16.558633)	-2377	0	0 1e-6	0 1e-6 (47.2, 155.2)	(14.7, 285.1)	(38.738600, 10.087623)	-2016	0	0 1e-6	0 1e-6 (50.7, 147.8)	(30.1, 101.1)

E.8 scene01-08/MER_FR__1PNUPA20031010_092717_000000502020_00351_08421_0453.N1

Warning : This product has not the expected number of tie-point lines.

DATA SET #3 - "Tie points ADS":

No.	Date	FIRST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	119093237.693105	(43.734696, 13.508266)	-26	0	0 1e-6	0 1e-6 (53.9, 153.3)	(18.4, 102.6)	(44.266045, 9.846784)	540	187	-665 1e-6	-5220 1e-6 (55.6, 149.3)	(38.0, 100.1)
9	119093260.219569	(42.410958, 13.104730)	1206	238	-786 1e-6	-4785 1e-6 (52.8, 152.5)	(18.4, 102.5)	(42.940479, 9.521488)	-128	0	0 1e-6	0 1e-6 (54.6, 148.5)	(38.0, 100.1)
17	119093282.746033	(41.086229, 12.712415)	-457	0	0 1e-6	0 1e-6 (51.8, 151.6)	(18.4, 102.5)	(41.614330, 9.202422)	442	65	-547 1e-6	-4095 1e-6 (53.5, 147.8)	(38.0, 100.1)
19	119093288.377649	(40.754899, 12.615973)	-2261	0	0 1e-6	0 1e-6 (51.5, 151.4)	(18.4, 102.4)	(41.282704, 9.123556)	-55	0	0 1e-6	0 1e-6 (53.3, 147.6)	(38.0, 100.1)

E.9 scene01-09/MER_FR__1PNUPA20031207_104236_000000982022_00180_09252_0537.N1

DATA SET #3 - "Tie points ADS":

No.	Date	FIRST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	124108956.659803	(49.653164, 7.216756)	409	50	1055 1e-6	4122 1e-6 (72.8, 170.4)	(37.9, 291.6)	(51.323409, -0.566129)	47	25	-8 1e-6	-46 1e-6 (75.5, 163.1)	(4.1, 105.6)
9	124108979.186267	(48.379342, 6.461337)	320	17	806 1e-6	3156 1e-6 (71.6, 169.7)	(37.9, 291.0)	(50.013974, -1.134740)	-60	0	0 1e-6	0 1e-6 (74.4, 162.6)	(4.1, 105.3)
17	124109001.712731	(47.100896, 5.741857)	250	13	615 1e-6	2415 1e-6 (70.4, 169.0)	(38.0, 290.5)	(48.702428, -1.678617)	-7	0	0 1e-6	0 1e-6 (73.2, 162.0)	(4.1, 105.0)
25	124109024.239195	(45.818209, 5.055281)	195	18	469 1e-6	1846 1e-6 (69.2, 168.3)	(38.0, 290.1)	(47.388935, -2.200081)	10	3	-1 1e-6	-9 1e-6 (72.1, 161.5)	(4.1, 104.8)
33	124109046.765659	(44.531627, 4.398887)	262	55	618 1e-6	2432 1e-6 (68.1, 167.7)	(38.0, 289.6)	(46.073645, -2.701183)	-104	0	0 1e-6	0 1e-6 (70.9, 161.0)	(4.1, 104.6)
36	124109055.213083	(44.048219, 4.160015)	137	34	320 1e-6	1263 1e-6 (67.6, 167.4)	(38.0, 289.5)	(45.579975, -2.884227)	-87	0	0 1e-6	0 1e-6 (70.5, 160.8)	(4.1, 104.5)

E.10 scene01-10/MER_FR__1PNUPA20031208_101129_000000982022_00194_09266_0535.N1

DATA SET #3 - "Tie points ADS":

No.	Date	FIRST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	124193489.202554	(49.271664, 8.360204)	106	4	32 1e-6	165 1e-6 (73.5, 163.9)	(6.7, 286.5)	(50.497857, 0.477256)	-44	0	0 1e-6	0 1e-6 (76.4, 156.7)	(36.1, 100.5)
9	124193511.729018	(47.965088, 7.780743)	324	157	96 1e-6	494 1e-6 (72.3, 163.4)	(6.7, 286.2)	(49.175646, 0.098816)	123	22	-146 1e-6	-1213 1e-6 (75.2, 156.3)	(36.1, 100.5)
17	124193534.255482	(46.656169, 7.225743)	1439	193	420 1e-6	2147 1e-6 (71.2, 162.8)	(6.7, 285.9)	(47.852658, -0.268492)	44	6	-52 1e-6	-423 1e-6 (74.1, 155.9)	(36.1, 100.4)
25	124193556.781946	(45.345087, 6.693028)	2153	390	619 1e-6	3142 1e-6 (70.0, 162.3)	(6.7, 285.7)	(46.528933, -0.625784)	106	15	-125 1e-6	-994 1e-6 (72.9, 155.5)	(36.1, 100.4)
33	124193579.308410	(44.032007, 6.180659)	747	231	211 1e-6	1067 1e-6 (68.9, 161.7)	(6.8, 285.4)	(45.204505, -0.974050)	21	2	-24 1e-6	-192 1e-6 (71.8, 155.1)	(36.1, 100.4)
36	124193587.755834	(43.539117, 5.993407)	319	51	89 1e-6	452 1e-6 (68.4, 161.5)	(6.8, 285.3)	(44.707670, -1.102505)	-5	0	0 1e-6	0 1e-6 (71.4, 155.0)	(36.1, 100.4)

E.11 scene02-01/MER_FR__1PNDPA20040302_103942_000000982024_00409_10483_1126.N1

DATA SET #3 - "Tie points ADS":

No.	Date	FIRST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	131539182.139727	(49.625284, 8.164910)	245	61	658 1e-6	2547 1e-6 (58.1, 162.5)	(38.8, 291.7)	(51.310939, 0.393494)	98	10	-11 1e-6	-64 1e-6 (61.5, 154.1)	(2.7, 105.8)
9	131539204.666191	(48.352861, 7.404165)	483	94	1267 1e-6	4915 1e-6 (57.0, 161.4)	(38.9, 291.2)	(50.002262, -0.181561)	-49	0	0 1e-6	0 1e-6 (60.4, 153.3)	(2.7, 105.5)
17	131539227.192655	(47.075725, 6.679666)	1044	177	2676 1e-6	10406 1e-6 (55.9, 160.4)	(38.9, 290.7)	(48.691410, -0.731444)	233	45	-25 1e-6	-144 1e-6 (59.4, 152.5)	(2.7, 105.2)
25	131539249.719119	(45.794270, 5.988361)	438	341	1098 1e-6	4279 1e-6 (54.9, 159.4)	(38.9, 290.2)	(47.378555, -1.258509)	39	19	-4 1e-6	-23 1e-6 (58.4, 151.6)	(2.7, 104.9)
33	131539272.245583	(44.508844, 5.327512)	826	254	2027 1e-6	7915 1e-6 (53.8, 158.4)	(38.9, 289.8)	(46.063849, -1.764842)	-40	0	0 1e-6	0 1e-6 (57.4, 150.8)	(2.7, 104.7)
36	131539280.693007	(44.025851, 5.087037)	173	144	421 1e-6	1646 1e-6 (53.4, 158.1)	(38.9, 289.6)	(45.570386, -1.949758)	-75	0	0 1e-6	0 1e-6 (57.0, 150.4)	(2.7, 104.6)

E.12 scene02-02/MER_FR__1PNDPA20040302_103942_000000982024_00409_10483_1130.N1

DATA SET #3 - "Tie points ADS":

No.	Date	FIRST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	131539182.139727	(51.145019, 1.308982)	59	0	7 1e-6	38 1e-6 (61.1, 155.1)	(2.7, 286.5)	(52.345710, -6.903788)	88	14	-110 1e-6	-1028 1e-6 (64.6, 146.7)	(38.8, 100.0)
9	131539204.666191	(49.839405, 0.710308)	14	17	1 1e-6	8 1e-6 (60.0, 154.2)	(2.7, 286.1)	(51.024262, -7.278522)	-97	0	0 1e-6	0 1e-6 (63.6, 146.0)	(38.9, 100.0)
17	131539227.192655	(48.531364, 0.138334)	248	9	28 1e-6	153 1e-6 (59.0, 153.4)	(2.7, 285.8)	(49.702075, -7.641864)	-127	0	0 1e-6	0 1e-6 (62.6, 145.4)	(38.9, 100.0)
25	131539249.719119	(47.221091, -0.409409)	60	13	6 1e-6	36 1e-6 (58.0, 152.5)	(2.7, 285.6)	(48.379184, -7.994999)	-147	0	0 1e-6	0 1e-6 (61.6, 144.7)	(38.9, 99.9)
33	131539272.245583	(45.908761, -0.935113)	15	12	1 1e-6	8 1e-6 (57.0, 151.6)	(2.7, 285.3)	(47.055620, -8.338972)	-4187	0	0 1e-6	0 1e-6 (60.6, 144.0)	(38.9, 99.9)
36	131539280.693007	(45.416139, -1.126975)	11	7	1 1e-6	6 1e-6 (56.6, 151.3)	(2.7, 285.2)	(46.559115, -8.465792)	-4784	0	0 1e-6	0 1e-6 (60.3, 143.7)	(38.9, 99.9)

E.13 scene02-03/MER_FR__1PNDPA20040302_104117_000000982024_00409_10483_1131.N1

DATA SET #3 - "Tie points ADS":

No.	Date	FIRST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt.	Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	131539277.874097	(44.187083, 5.166854)	619	265	1511 1e-6	5903 1e-6 (53.6, 158.2)	(38.9, 289.7)	(45.735081, -1.888340)	-62	0	0 1e-6	0 1e-6 (57.1, 150.5)	(2.7, 104.7)
9	131539300.400561	(42.897134, 4.540645)	-525	0	0 1e-6	0 1e-6 (52.5, 157.2)	(38.9, 289.2)	(44.418253, -2.371344)	-1344	0	0 1e-6	0 1e-6 (56.1, 149.7)	(2.7, 104.5)
17	131539322.927025	(41.603879, 3.939703)	-2125	0	0 1e-6	0 1e-6 (51.5, 156.2)	(38.9, 288.8)	(43.099862, -2.837508)	649	186	-68 1e-6	-367 1e-6 (55.1, 148.8)	(2.7, 104.3)
25	131539345.453489	(40.307565, 3.362081)	-1509	0	0 1e-6	0 1e-6 (50.5, 155.1)	(38.9, 288.5)	(41.780015, -3.288272)	1001	25	-104 1e-6	-555 1e-6 (54.1, 147.9)	(2.7, 104.1)
33	131539367.979953	(39.008419, 2.806016)	-166	0	0 1e-6	0 1e-6 (49.4, 154.1)	(38.9, 288.1)	(40.458810, -3.724928)	681	22	-70 1e-6	-370 1e-6 (53.2, 147.0)	(2.7, 104.0)
36	131539376.427377	(38.520551, 2.602718)	-2043	0	0 1e-6	0 1e-6 (49.0, 153.7)	(38.9, 288.0)	(39.963026, -3.885279)	472	41	-48 1e-6	-254 1e-6 (52.8, 146.6)	(2.7, 103.9)

E.14 scene02-04/MER_FR__1PNPDE20040328_102200_000000982025_00280_10855_0002.N1

DATA SET #3 - "Tie points ADS":

No.	Date	FIRST (Lat., Lon.)	Alt. Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt. Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	133784520.248060	(51.198988, 13.715287)	164 32	473 1e-6	1807 1e-6	(49.0, 164.0)	(39.7, 292.7)	(52.950155, 5.704595)	-2 0	0 1e-6	0 1e-6	(52.4, 154.4)	(1.4, 106.4)
9	133784542.774524	(49.934628, 12.898227)	651 37	1830 1e-6	7017 1e-6	(48.0, 162.7)	(39.7, 292.1)	(51.645319, 5.087028)	10 1	0 1e-6	-3 1e-6	(51.4, 153.3)	(1.4, 106.0)
17	133784565.300988	(48.664890, 12.122121)	464 26	1273 1e-6	4895 1e-6	(46.9, 161.5)	(39.8, 291.5)	(50.337982, 4.498547)	175 20	-10 1e-6	-56 1e-6	(50.4, 152.2)	(1.4, 105.7)
25	133784587.827452	(47.390239, 11.383432)	1809 613	4851 1e-6	18691 1e-6	(45.8, 160.2)	(39.8, 291.0)	(49.028355, 3.936351)	120 33	-6 1e-6	-37 1e-6	(49.4, 151.1)	(1.4, 105.4)
33	133784610.353916	(46.111086, 10.679000)	1503 319	3941 1e-6	15220 1e-6	(44.8, 158.9)	(39.8, 290.5)	(47.716621, 3.397969)	247 34	-13 1e-6	-75 1e-6	(48.4, 150.0)	(1.4, 105.2)
36	133784618.801340	(45.630319, 10.423082)	404 46	1050 1e-6	4061 1e-6	(44.4, 158.4)	(39.8, 290.3)	(47.224212, 3.201762)	254 26	-14 1e-6	-76 1e-6	(48.0, 149.6)	(1.4, 105.1)

E.15 MER_RR__1PNPDK20030408_093303_000023202015_00208_05773_0848.N1

This product has been generated on February 2003 and contains a defect that has been fixed in the processor: Latitude and longitude corrections are not expressed in $1e^{-6}$ degrees as stated in document R-2 but in $1e^{-6}$ radians.

DATA SET #3 - "Tie points ADS"

No.	Date	FIRST (Lat., Lon.)	Alt. Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)	LAST (Lat., Lon.)	Alt. Rough.	Corr. Lat. Lon.	Sun (Zen., Azi.)	View (Zen., Azi.)
1	103109583.820036	(69.010207, 47.018145)	-51 0	-4 1e-6	-14 1e-6	(62.2, 191.0)	(40.5, 311.8)	(74.042619, 17.620962)	-242 0	7 1e-6	114 1e-6	(67.9, 158.9)	(40.5, 103.8)
9	103109606.346500	(67.975274, 44.591548)	20 11	1 1e-6	5 1e-6	(61.1, 188.4)	(40.5, 309.6)	(72.747260, 16.535418)	-437 0	13 1e-6	192 1e-6	(66.8, 157.7)	(40.5, 103.1)
17	103109628.872964	(66.908355, 42.372819)	-7 1	0 1e-6	-1 1e-6	(59.9, 186.1)	(40.5, 307.6)	(71.447998, 15.575876)	-1293 0	37 1e-6	532 1e-6	(65.7, 156.5)	(40.5, 102.5)
25	103109651.399428	(65.813653, 40.339576)	-8 27	0 1e-6	-2 1e-6	(58.7, 183.9)	(40.5, 305.8)	(70.145543, 14.717798)	-2501 0	69 1e-6	966 1e-6	(64.6, 155.4)	(40.5, 102.0)
33	103109673.925892	(64.694747, 38.471623)	-1 16	0 1e-6	0 1e-6	(57.6, 181.9)	(40.5, 304.2)	(68.840428, 13.942550)	-91 0	2 1e-6	33 1e-6	(63.5, 154.4)	(40.5, 101.5)
41	103109696.452356	(63.554680, 36.750928)	180 30	13 1e-6	45 1e-6	(56.4, 180.0)	(40.5, 302.7)	(67.533060, 13.235743)	-238 0	6 1e-6	82 1e-6	(62.5, 153.5)	(40.5, 101.2)
49	103109718.978820	(62.396046, 35.161501)	60 23	4 1e-6	14 1e-6	(55.3, 178.2)	(40.5, 301.3)	(66.223758, 12.586091)	-71 75	1 1e-6	23 1e-6	(61.4, 152.5)	(40.5, 100.9)
57	103109741.505284	(61.221061, 33.689225)	120 19	8 1e-6	28 1e-6	(54.1, 176.5)	(40.5, 300.0)	(64.912769, 11.984627)	233 77	-5 1e-6	-72 1e-6	(60.3, 151.6)	(40.5, 100.6)
65	103109764.031748	(60.031621, 32.321670)	10 9	0 1e-6	2 1e-6	(53.0, 174.8)	(40.5, 298.9)	(63.600293, 11.424131)	442 65	-10 1e-6	-131 1e-6	(59.3, 150.7)	(40.5, 100.4)
73	103109786.558212	(58.829354, 31.047899)	50 6	3 1e-6	11 1e-6	(51.9, 173.2)	(40.6, 297.8)	(62.286487, 10.898731)	647 83	-15 1e-6	-184 1e-6	(58.3, 149.8)	(40.6, 100.2)
81	103109809.084676	(57.615663, 29.858292)	64 11	3 1e-6	14 1e-6	(50.7, 171.7)	(40.6, 296.9)	(60.971483, 10.403599)	635 88	-14 1e-6	-173 1e-6	(57.2, 149.0)	(40.6, 100.1)
89	103109831.611140	(56.391762, 28.744384)	148 11	8 1e-6	32 1e-6	(49.6, 170.3)	(40.6, 296.0)	(59.655387, 9.934729)	195 92	-4 1e-6	-51 1e-6	(56.2, 148.1)	(40.6, 100.0)
97	103109854.312018	(55.149121, 27.690865)	173 5	9 1e-6	36 1e-6	(48.5, 168.8)	(40.6, 295.1)	(58.328084, 9.485400)	-635 0	14 1e-6	160 1e-6	(55.2, 147.2)	(40.6, 99.9)
105	103109876.838482	(53.907758, 26.707302)	207 12	11 1e-6	43 1e-6	(47.4, 167.4)	(40.6, 294.4)	(57.010045, 9.059673)	4 0	0 1e-6	0 1e-6	(54.2, 146.3)	(40.6, 99.8)
113	103109899.364946	(52.658958, 25.779529)	148 5	7 1e-6	30 1e-6	(46.3, 166.0)	(40.6, 293.6)	(55.691137, 8.651612)	25 4	0 1e-6	-5 1e-6	(53.2, 145.4)	(40.6, 99.7)
121	103109921.891410	(51.403423, 24.902395)	166 7	8 1e-6	33 1e-6	(45.2, 164.6)	(40.6, 293.0)	(54.371413, 8.259167)	-12 0	0 1e-6	2 1e-6	(52.2, 144.5)	(40.6, 99.7)
129	103109944.417874	(50.141770, 24.071324)	221 14	11 1e-6	42 1e-6	(44.1, 163.2)	(40.6, 292.4)	(53.050918, 7.880573)	12 4	0 1e-6	-2 1e-6	(51.2, 143.5)	(40.6, 99.6)
137	103109966.944338	(48.874542, 23.282246)	788 90	39 1e-6	149 1e-6	(43.0, 161.9)	(40.6, 291.8)	(51.729692, 7.514298)	80 13	-1 1e-6	-17 1e-6	(50.3, 142.5)	(40.6, 99.6)
145	103109989.470802	(47.602224, 22.531532)	127 10	6 1e-6	23 1e-6	(42.0, 160.5)	(40.6, 291.2)	(50.407769, 7.159013)	467 47	-10 1e-6	-97 1e-6	(49.3, 141.5)	(40.6, 99.6)
153	103110011.997266	(46.325245, 21.815938)	155 132	7 1e-6	28 1e-6	(40.9, 159.1)	(40.6, 290.7)	(49.085180, 6.813549)	268 16	-6 1e-6	-54 1e-6	(48.4, 140.5)	(40.6, 99.6)
161	103110034.523730	(45.043987, 21.132555)	103 22	4 1e-6	18 1e-6	(39.9, 157.6)	(40.7, 290.3)	(47.761952, 6.476881)	335 12	-7 1e-6	-66 1e-6	(47.5, 139.4)	(40.7, 99.6)
169	103110057.050194	(43.758792, 20.478773)	324 116	14 1e-6	56 1e-6	(38.9, 156.2)	(40.7, 289.8)	(46.438111, 6.148100)	1288 259	-29 1e-6	-248 1e-6	(46.6, 138.3)	(40.7, 99.6)
177	103110079.576658	(42.469970, 19.852244)	1694 429	75 1e-6	292 1e-6	(37.9, 154.7)	(40.7, 289.4)	(45.113678, 5.826400)	496 410	-11 1e-6	-93 1e-6	(45.7, 137.2)	(40.7, 99.7)
185	103110102.103122	(41.177797, 19.250847)	-85 0	-3 1e-6	-14 1e-6	(36.9, 153.2)	(40.7, 289.0)	(43.788677, 5.511059)	541 48	-12 1e-6	-99 1e-6	(44.9, 136.0)	(40.7, 99.7)
193	103110124.632316	(39.882367, 18.672597)	-128 0	-5 1e-6	-21 1e-6	(35.9, 151.6)	(40.7, 288.6)	(42.462966, 5.201393)	-1915 0	43 1e-6	345 1e-6	(44.0, 134.8)	(40.7, 99.7)
201	103110147.158780	(38.584220, 18.115900)	-2583 0	-109 1e-6	-424 1e-6	(34.9, 150.0)	(40.7, 288.3)	(41.136887, 4.896896)	-2581 0	59 1e-6	456 1e-6	(43.2, 133.6)	(40.7, 99.8)
209	103110169.685244	(37.283408, 17.579107)	-2866 0	-119 1e-6	-463 1e-6	(34.0, 148.3)	(40.7, 287.9)	(39.810298, 4.597003)	-792 0	18 1e-6	137 1e-6	(42.4, 132.3)	(40.7, 99.8)
217	103110192.211708	(35.980119, 17.060785)	-3596 0	-147 1e-6	-572 1e-6	(33.1, 146.5)	(40.7, 287.6)	(38.483217, 4.301234)	-2584 0	59 1e-6	439 1e-6	(41.7, 130.9)	(40.7, 99.9)
225	103110214.738172	(34.674525, 16.559626)	-1544 0	-62 1e-6	-242 1e-6	(32.2, 144.7)	(40.7, 287.3)	(37.155664, 4.009153)	-2339 0	54 1e-6	391 1e-6	(40.9, 129.5)	(40.7, 99.9)
233	103110237.264636	(33.366786, 16.074438)	-1212 0	-48 1e-6	-187 1e-6	(31.3, 142.8)	(40.7, 287.0)	(35.827656, 3.720359)	664 20	-15 1e-6	-109 1e-6	(40.2, 128.1)	(40.7, 100.0)
241	103110259.791100	(32.057047, 15.604127)	-64 0	-2 1e-6	-9 1e-6	(30.5, 140.9)	(40.7, 286.7)	(34.499212, 3.434483)	1038 32	-24 1e-6	-167 1e-6	(39.5, 126.6)	(40.7, 100.0)
249	103110282.317564	(30.745443, 15.147690)	152 22	5 1e-6	22 1e-6	(29.7, 138.8)	(40.8, 286.5)	(33.170350, 3.151185)	755 9	-17 1e-6	-120 1e-6	(38.9, 125.1)	(40.8, 100.1)

257	103110304.844028	(29.432101,	14.704209)	344	17	13 1e-6	51 1e-6	(28.9, 136.7)	(40.8, 286.2)	(31.841091,	2.870148)	689	18	-16 1e-6	-108 1e-6	(38.3, 123.5)	(40.8, 100.2)
265	103110327.370492	(28.117135,	14.272835)	546	18	20 1e-6	80 1e-6	(28.2, 134.4)	(40.8, 286.0)	(30.511453,	2.591077)	453	7	-10 1e-6	-70 1e-6	(37.7, 121.8)	(40.8, 100.3)
273	103110349.896956	(26.800654,	13.852787)	504	21	18 1e-6	73 1e-6	(27.5, 132.1)	(40.8, 285.8)	(29.181457,	2.313696)	471	12	-11 1e-6	-71 1e-6	(37.1, 120.2)	(40.8, 100.3)
281	103110372.423420	(25.482760,	13.443343)	646	3	23 1e-6	93 1e-6	(26.8, 129.6)	(40.8, 285.5)	(27.851122,	2.037744)	529	23	-12 1e-6	-79 1e-6	(36.6, 118.4)	(40.8, 100.4)
289	103110394.948708	(24.163617,	13.043857)	717	6	25 1e-6	102 1e-6	(26.2, 127.1)	(40.8, 285.3)	(26.520540,	1.762992)	152	2	-3 1e-6	-22 1e-6	(36.1, 116.6)	(40.8, 100.5)
297	103110417.475172	(22.843176,	12.653666)	767	12	27 1e-6	108 1e-6	(25.7, 124.4)	(40.8, 285.1)	(25.189593,	1.489176)	305	26	-7 1e-6	-44 1e-6	(35.7, 114.8)	(40.8, 100.6)
305	103110440.001636	(21.521591,	12.272217)	719	4	25 1e-6	101 1e-6	(25.2, 121.7)	(40.8, 284.9)	(23.858374,	1.216089)	306	7	-7 1e-6	-44 1e-6	(35.3, 112.9)	(40.8, 100.7)
313	103110462.528100	(20.198940,	11.898976)	400	6	13 1e-6	55 1e-6	(24.7, 118.9)	(40.8, 284.8)	(22.526905,	0.943519)	360	1	-9 1e-6	-51 1e-6	(35.0, 111.0)	(40.8, 100.7)
321	103110485.054564	(18.875300,	11.533442)	555	3	18 1e-6	77 1e-6	(24.4, 115.9)	(40.8, 284.6)	(21.195211,	0.671259)	382	3	-9 1e-6	-54 1e-6	(34.7, 109.1)	(40.8, 100.8)
329	103110507.581028	(17.550742,	11.175152)	411	0	13 1e-6	56 1e-6	(24.0, 112.9)	(40.8, 284.4)	(19.863315,	0.399112)	381	2	-9 1e-6	-53 1e-6	(34.4, 107.1)	(40.8, 100.9)
337	103110530.107492	(16.225334,	10.823668)	397	6	13 1e-6	54 1e-6	(23.8, 109.9)	(40.8, 284.2)	(18.531245,	0.126886)	403	2	-10 1e-6	-56 1e-6	(34.2, 105.1)	(40.8, 101.0)
345	103110552.633956	(14.899140,	10.478584)	437	4	14 1e-6	59 1e-6	(23.6, 106.8)	(40.8, 284.1)	(17.199026,	-0.145608)	324	3	-8 1e-6	-45 1e-6	(34.1, 103.1)	(40.8, 101.1)
353	103110575.160420	(13.572225,	10.139516)	465	12	15 1e-6	63 1e-6	(23.5, 103.6)	(40.8, 283.9)	(15.866684,	-0.418554)	386	9	-10 1e-6	-53 1e-6	(33.9, 101.0)	(40.8, 101.2)
361	103110597.686884	(12.244646,	9.806103)	361	5	11 1e-6	48 1e-6	(23.4, 100.5)	(40.8, 283.8)	(14.534250,	-0.692135)	302	12	-8 1e-6	-41 1e-6	(33.9, 99.0)	(40.8, 101.3)
369	103110620.213348	(10.916460,	9.478006)	610	24	19 1e-6	81 1e-6	(23.5, 97.3)	(40.8, 283.6)	(13.201750,	-0.966529)	303	12	-8 1e-6	-41 1e-6	(33.9, 96.9)	(40.8, 101.4)
377	103110642.739812	(9.587724,	9.154905)	1139	12	35 1e-6	152 1e-6	(23.5, 94.2)	(40.8, 283.5)	(11.869216,	-1.241915)	273	0	-7 1e-6	-37 1e-6	(33.9, 94.9)	(40.8, 101.5)
385	103110665.269006	(8.258327,	8.836457)	170	6	5 1e-6	22 1e-6	(23.7, 91.1)	(40.8, 283.3)	(10.536516,	-1.518506)	152	19	-4 1e-6	-20 1e-6	(34.0, 92.9)	(40.8, 101.6)
393	103110687.795470	(6.928644,	8.522453)	152	4	4 1e-6	20 1e-6	(23.9, 88.0)	(40.8, 283.2)	(9.204005,	-1.796414)	178	6	-4 1e-6	-23 1e-6	(34.1, 90.9)	(40.8, 101.7)
401	103110710.321934	(5.598562,	8.212579)	152	14	4 1e-6	20 1e-6	(24.2, 85.1)	(40.8, 283.0)	(7.871554,	-2.075851)	229	23	-6 1e-6	-30 1e-6	(34.3, 88.9)	(40.8, 101.8)
409	103110732.848398	(4.268129,	7.906576)	-4	0	0 1e-6	0 1e-6	(24.6, 82.2)	(40.8, 282.9)	(6.539196,	-2.357001)	146	77	-4 1e-6	-19 1e-6	(34.6, 86.9)	(40.8, 101.9)
417	103110755.374862	(2.937389,	7.604195)	-2314	0	-69 1e-6	-306 1e-6	(25.0, 79.4)	(40.8, 282.8)	(5.206967,	-2.640048)	41	1	-1 1e-6	-5 1e-6	(34.8, 85.0)	(40.8, 102.1)
425	103110777.901326	(1.606388,	7.305199)	-1644	0	-48 1e-6	-217 1e-6	(25.5, 76.7)	(40.8, 282.7)	(3.874902,	-2.925181)	-2916	0	83 1e-6	387 1e-6	(35.2, 83.1)	(40.8, 102.2)
433	103110800.427790	(0.275169,	7.009358)	-2809	0	-82 1e-6	-371 1e-6	(26.0, 74.2)	(40.8, 282.5)	(2.543036,	-3.212592)	-4979	0	143 1e-6	660 1e-6	(35.5, 81.3)	(40.8, 102.3)
441	103110822.954254	(-1.056227,	6.716454)	-3197	0	-93 1e-6	-423 1e-6	(26.6, 71.7)	(40.8, 282.4)	(1.211409,	-3.502478)	-5006	0	145 1e-6	662 1e-6	(36.0, 79.5)	(40.8, 102.4)
449	103110845.480718	(-2.387761,	6.426274)	-3625	0	-104 1e-6	-480 1e-6	(27.2, 69.4)	(40.8, 282.3)	(-0.119941,	-3.795045)	-5006	0	147 1e-6	662 1e-6	(36.4, 77.8)	(40.8, 102.5)
457	103110868.007182	(-3.719394,	6.138614)	-4544	0	-129 1e-6	-603 1e-6	(27.9, 67.1)	(40.8, 282.2)	(-1.450976,	-4.090501)	-5000	0	148 1e-6	661 1e-6	(36.9, 76.1)	(40.8, 102.6)
465	103110890.533646	(-5.051087,	5.853275)	-4817	0	-136 1e-6	-640 1e-6	(28.6, 65.0)	(40.8, 282.1)	(-2.781653,	-4.389063)	-4803	0	143 1e-6	635 1e-6	(37.4, 74.5)	(40.8, 102.8)
473	103110913.060110	(-6.382805,	5.570063)	-4885	0	-137 1e-6	-651 1e-6	(29.4, 63.0)	(40.8, 282.0)	(-4.111930,	-4.690959)	-4402	0	133 1e-6	582 1e-6	(38.0, 72.9)	(40.8, 102.9)
481	103110935.585398	(-7.714442,	5.288802)	-5090	0	-141 1e-6	-680 1e-6	(30.1, 61.2)	(40.7, 281.8)	(-5.441695,	-4.996407)	-4740	0	144 1e-6	627 1e-6	(38.6, 71.4)	(40.7, 103.0)
489	103110958.111862	(-9.046103,	5.009279)	-5289	0	-145 1e-6	-709 1e-6	(31.0, 59.4)	(40.7, 281.7)	(-6.771040,	-5.305685)	-4410	0	135 1e-6	585 1e-6	(39.3, 70.0)	(40.7, 103.2)
497	103110980.638326	(-10.377686,	4.731325)	-5416	0	-147 1e-6	-729 1e-6	(31.8, 57.7)	(40.7, 281.6)	(-8.099851,	-5.619034)	-4600	0	143 1e-6	611 1e-6	(39.9, 68.6)	(40.7, 103.3)
505	103111003.164790	(-11.709158,	4.454759)	-5467	0	-147 1e-6	-739 1e-6	(32.7, 56.1)	(40.7, 281.5)	(-9.428079,	-5.936724)	-4299	0	135 1e-6	573 1e-6	(40.6, 67.3)	(40.7, 103.4)
513	103111025.691254	(-13.040488,	4.179403)	-5429	0	-145 1e-6	-738 1e-6	(33.6, 54.6)	(40.7, 281.4)	(-10.755675,	-6.259039)	-4438	0	140 1e-6	593 1e-6	(41.4, 66.0)	(40.7, 103.6)
521	103111048.217718	(-14.371647,	3.905077)	-5434	0	-144 1e-6	-743 1e-6	(34.5, 53.2)	(40.7, 281.3)	(-12.082587,	-6.586279)	-4405	0	141 1e-6	591 1e-6	(42.1, 64.8)	(40.7, 103.7)
529	103111070.744182	(-15.702605,	3.631604)	-5437	0	-142 1e-6	-748 1e-6	(35.5, 51.9)	(40.7, 281.2)	(-13.408761,	-6.918758)	-3984	0	129 1e-6	536 1e-6	(42.9, 63.7)	(40.7, 103.9)
537	103111093.270646	(-17.033335,	3.358804)	-5403	0	-140 1e-6	-748 1e-6	(36.4, 50.7)	(40.7, 281.1)	(-14.734143,	-7.256813)	-4198	0	137 1e-6	568 1e-6	(43.7, 62.6)	(40.7, 104.0)
545	103111115.797110	(-18.363809,	3.086495)	-5407	0	-139 1e-6	-754 1e-6	(37.4, 49.5)	(40.6, 281.0)	(-16.058674,	-7.600796)	-4331	0	143 1e-6	589 1e-6	(44.6, 61.6)	(40.6, 104.2)
553	103111138.323574	(-19.694000,	2.814495)	-5376	0	-137 1e-6	-755 1e-6	(38.4, 48.5)	(40.6, 280.9)	(-17.382294,	-7.951087)	-4194	0	140 1e-6	574 1e-6	(45.4, 60.6)	(40.6, 104.4)
561	103111160.850038	(-21.023885,	2.542617)	-5388	0	-136 1e-6	-764 1e-6	(39.4, 47.4)	(40.6, 280.9)	(-18.704940,	-8.308085)	-3829	0	129 1e-6	527 1e-6	(46.3, 59.7)	(40.6, 104.5)
569	103111183.376502	(-22.353439,	2.270668)	-5181	0	-130 1e-6	-741 1e-6	(40.5, 46.5)	(40.6, 280.8)	(-20.026546,	-8.672218)	-3715	0	127 1e-6	515 1e-6	(47.2, 58.8)	(40.6, 104.7)
577	103111205.905696	(-23.682798,	1.998420)	-5244	0	-130 1e-6	-757 1e-6	(41.5, 45.6)	(40.6, 280.7)	(-21.347203,	-9.043991)	-3992	0	138 1e-6	557 1e-6	(48.1, 58.0)	(40.6, 104.9)
585	103111228.432160	(-25.011619,	1.725736)	-5155	0	-127 1e-6	-752 1e-6	(42.6, 44.7)	(40.6, 280.6)	(-22.666518,	-9.423801)	-4085	0	143 1e-6	575 1e-6	(49.0, 57.3)	(40.6, 105.1)
593	103111250.958624	(-26.340042,	1.452371)	-5241	0	-128 1e-6	-773 1e-6	(43.6, 43.9)	(40.6, 280.5)	(-23.984574,	-9.812219)	-4182	0	148 1e-6	593 1e-6	(50.0, 56.6)	(40.6, 105.3)
601	103111273.485088	(-27.668044,	1.178107)	-4589	0	-111 1e-6	-684 1e-6	(44.7, 43.2)	(40.5, 280.4)	(-25.301289,	-10.209809)	-4038	0	145 1e-6	578 1e-6	(50.9, 55.9)	(40.5, 105.5)
609	103111296.011552	(-28.995607,	0.902712)	-4421	0	-106 1e-6	-667 1e-6	(45.8, 42.5)	(40.5, 280.3)	(-26.616578,	-10.617177)	-3976	0	144 1e-6	574 1e-6	(51.9, 55.3)	(40.5, 105.7)
617	103111318.538016	(-30.322710,	0.625946)	-2517	0	-60 1e-6	-385 1e-6	(46.9, 41.9)	(40.5, 280.3)	(-27.930350,	-11.034978)	-3569	0	131 1e-6	521 1e-6	(52.9, 54.8)	(40.5, 106.0)
625	103111341.064480	(-31.649335,	0.347551)	-4504	0	-106 1e-6	-698 1e-6	(48.0, 41.2)	(40.5, 280.2)	(-29.242505,	-11.463918)	-3003	0	112 1e-6	443 1e-6	(53.9, 54.3)	(40.5, 106.2)
633	103111363.590944	(-32.975464,	0.067257)	-2615	0	-61 1e-6	-411 1e-6	(49.1, 40.7)	(40.5, 280.1)	(-30.552940,	-11.904761)	-3593	0	136 1e-6	536 1e-6	(54.9, 53.8)	(40.5, 106.4)
641	103111386.117408	(-34.301080,	-0.215225)	-4596	0	-107 1e-6	-733 1e-6	(50.2, 40.2)	(40.5, 280.1)	(-31.861543,	-12.358337)	-3451	0	132 1e-6	521 1e-6	(55.9, 53.4)	(40.5, 106.7)
649	103111408.643872	(-35.626165,	-0.500208)	-4842	0	-112 1e-6	-785 1e-6	(51.3, 39.7)	(40.4, 280.0)	(-33.168194,	-12.825543)	-3134	0	122 1e-6	479 1e-6	(57.0, 53.0)	(40.4, 107.0)
657	103111431.170336	(-36.950703,	-0.788024)	-5002	0	-115 1e-6	-824 1e-6	(52.4, 39.2)	(40.4, 279.9)	(-34.472761,	-13.307359)	-3626	0	143 1e-6	562 1e-6	(58.0, 52.7)	(40.4, 107.3)
665	103111453.696800	(-38.274677,	-1.079036)	-5025	0	-115 1e-6	-843 1e-6	(53.5, 38.8)	(40.4, 279.9)	(-35.775105,	-13.804851)	-3494	0	140 1e-6	549 1e-6	(59.1, 52.4)	(40.4, 107.6)
673	103111476.225994	(-39.598232,	-1.373669)	-4887	0	-111 1e-6	-835 1e-6	(54.7, 38.4)	(40.4, 279.8)	(-37.075231,	-14.319246)	-3005	0	123 1e-6	479 1e-6	(60.1, 52.2)	(40.4, 107.9)
681	103111498.752458	(-40.921031,	-1.672279)	-4836	0	-109 1e-6	-842 1e-6	(55.8, 38.1)	(40.4, 279.8)	(-38.372656,	-14.851694)	-3001	0	125 1e-6	485 1e-6	(61.2, 52.0)	(40.4, 108.2)
689	103111521.278922	(-42.243219,	-1.975369)	-4179	0	-94 1e-6	-742 1e-6	(57.0, 37.7)	(40.3, 279.7)	(-39.667357,	-15.403651)	-3021	0	128 1e-6	496 1e-6	(62.3, 51.9)	(40.3, 108.6)
697	103111543.805386	(-43.564778,	-2.283451)	-4231	0	-94 1e-6	-767 1e-6	(58.1, 37.5)	(40.3, 279.7)	(-40.959135,	-15.976651)	-3015	0	130 1e-6	503 1e-6	(63.4, 51.8)	(40.3, 108.9)
705	103111566.331850	(-44.885692,	-2.597094)	-4021	0	-89 1e-6	-745 1e-6	(59.2, 37.2)	(40.3, 279.7)	(-42.247771,	-16.572383)	-2979	0	131 1e-6	506 1e-6	(64.5, 51.7)	(40.3, 109.3)
713	103111588.858314	(-46.205944,	-2.916923)	-4186	0	-93 1e-6	-794 1e-6	(60.4, 37.0)	(40.3, 279.6)	(-43.533023,	-17.192706)	-3027	0	136 1e-6	523 1e-6	(65.6, 51.7)	(40.3, 109.7)

```
721  103111611.384778  (-47.525514,  -3.243639 ) -4239    0  -94 1e-6  -823 1e-6 ( 61.5,  36.8 )  ( 40.3, 279.6 )  (-44.814626, -17.839679 ) -3510    0  161 1e-6   618 1e-6 ( 66.7,  51.8 )  ( 40.3, 110.2 )
729  103111633.911242  (-48.844383,  -3.578024 ) -3947    0  -87 1e-6  -786 1e-6 ( 62.7,  36.6 )  ( 40.2, 279.6 )  (-46.092284, -18.515580 ) -3976    0  186 1e-6   713 1e-6 ( 67.8,  51.9 )  ( 40.2, 110.6 )
737  103111656.437706  (-50.162529,  -3.920955 ) -2962    0  -65 1e-6  -606 1e-6 ( 63.9,  36.5 )  ( 40.2, 279.6 )  (-47.365666, -19.222939 ) -4000    0  191 1e-6   732 1e-6 ( 68.9,  52.0 )  ( 40.2, 111.1 )
745  103111678.964170  (-51.479928,  -4.273426 ) -2400    0  -53 1e-6  -504 1e-6 ( 65.0,  36.4 )  ( 40.2, 279.6 )  (-48.634404, -19.964570 ) -3326    0  163 1e-6   621 1e-6 ( 70.0,  52.2 )  ( 40.2, 111.7 )
753  103111701.490634  (-52.796553,  -4.636560 ) -2033    0  -45 1e-6  -440 1e-6 ( 66.2,  36.3 )  ( 40.2, 279.6 )  (-49.898088, -20.743613 ) -4096    0  205 1e-6   781 1e-6 ( 71.2,  52.5 )  ( 40.2, 112.2 )
761  103111724.017098  (-54.112373,  -5.011641 ) -2463    0  -54 1e-6  -549 1e-6 ( 67.3,  36.2 )  ( 40.2, 279.7 )  (-51.156253, -21.563575 ) -4540    0  233 1e-6   885 1e-6 ( 72.3,  52.8 )  ( 40.2, 112.9 )
769  103111746.718167  (-55.437543,  -5.403206 ) -2992    0  -66 1e-6  -689 1e-6 ( 68.5,  36.2 )  ( 40.2, 279.7 )  (-52.418057, -22.435275 ) -4501    0  238 1e-6   897 1e-6 ( 73.4,  53.2 )  ( 40.2, 113.5 )
777  103111769.244631  (-56.751636,  -5.806939 ) -3104    0  -69 1e-6  -739 1e-6 ( 69.7,  36.2 )  ( 40.1, 279.8 )  (-53.663494, -23.349747 ) -4018    0  218 1e-6   819 1e-6 ( 74.6,  53.7 )  ( 40.1, 114.2 )
785  103111791.771095  (-58.064801,  -6.227764 ) -4006    0  -90 1e-6  -988 1e-6 ( 70.8,  36.3 )  ( 40.1, 279.8 )  (-54.901622, -24.318486 ) -3991    0  223 1e-6   833 1e-6 ( 75.7,  54.2 )  ( 40.1, 115.0 )
793  103111814.297559  (-59.376981,  -6.667977 ) -4999    0 -114 1e-6 -1279 1e-6 ( 72.0,  36.3 )  ( 40.1, 279.9 )  (-56.131674, -25.347094 ) -7005    0  403 1e-6  1498 1e-6 ( 76.9,  54.8 )  ( 40.1, 115.8 )
801  103111836.824023  (-60.688109,  -7.130273 ) -5220    0 -120 1e-6 -1389 1e-6 ( 73.2,  36.5 )  ( 40.1, 280.1 )  (-57.352774, -26.441906 ) -1893    0  112 1e-6   414 1e-6 ( 78.0,  55.5 )  ( 40.1, 116.7 )
809  103111859.350487  (-61.998105,  -7.617840 ) -5204    0 -121 1e-6 -1442 1e-6 ( 74.3,  36.6 )  ( 40.1, 280.2 )  (-58.563918, -27.610088 ) -3209    0  196 1e-6   721 1e-6 ( 79.2,  56.2 )  ( 40.1, 117.6 )
817  103111881.876951  (-63.306870,  -8.134481 ) -5228    0 -124 1e-6 -1513 1e-6 ( 75.5,  36.8 )  ( 40.1, 280.4 )  (-59.763948, -28.859770 ) -2518    0  159 1e-6   580 1e-6 ( 80.3,  57.1 )  ( 40.1, 118.6 )
825  103111904.403415  (-64.614286,  -8.684763 ) -5193    0 -125 1e-6 -1573 1e-6 ( 76.7,  37.1 )  ( 40.1, 280.6 )  (-60.951527, -30.200177 ) -4133    0  271 1e-6   976 1e-6 ( 81.5,  58.1 )  ( 40.1, 119.8 )
```

APPENDIX F - QUALITY CONTROLS OF ACE AND GETASSE30

F.1 Differences between GPS points and ACE DEM – Nearest neighbour

```
DEM "../DEM_ACE/ACE_30sec/EUROPA_WGS84" controlled by "EUVN_ETRF.BLH" GCPs:
NUMBER OF GCPs IN INPUT          :      192
NUMBER OF PROCESSED GCPs         :      144 (75.0%)
NUMBER OF GCPs OUTSITE THE DEM   :       48 (25.0%)
ELEVATION DIFFERENCES :
. ARITHMETIC MEAN                :    -2201.149
. QUADRATIC  MEAN                :    12109.911
. STANDARD DEVIATION             :    11908.186
. MINIMUM                        :   -65001.139
. MAXIMUM                        :      325.560
```

GCP #	EASTING	NORTHING	GCP ELEVAT.	DEM ELEVAT.	DIFFERENCE						
1 :	17.073455	52.276957	124.366	78.000 ->	46.366	21 :	-6.205649	36.464351	84.180	51.000 ->	33.180
2 :	4.359235	50.797807	149.668	101.000 ->	48.668	22 :	11.877930	45.406732	84.043	16.000 ->	68.043
3 :	8.972749	39.135881	238.378	142.000 ->	96.378	23 :	-3.951976	40.443603	647.362	595.000 ->	52.362
4 :	3.399645	50.933714	63.894	15.000 ->	48.894	24 :	18.367308	57.653880	79.787	31.000 ->	48.787
5 :	4.594950	50.094902	282.695	148.000 ->	134.695	25 :	12.878888	49.144195	666.026	587.000 ->	79.026
6 :	0.492343	40.820895	107.810	6.000 ->	101.810	26 :	7.465296	46.877115	956.342	850.000 ->	106.342
7 :	14.785635	49.913677	592.594	466.000 ->	126.594	27 :	15.598375	48.654545	457.606	393.000 ->	64.606
8 :	6.920585	43.754725	1319.310	1260.000 ->	59.310	28 :	13.683435	46.554197	687.372	568.000 ->	119.372
9 :	15.493466	47.067107	538.290	430.000 ->	108.290	29 :	12.082992	47.496026	630.059	531.000 ->	99.059
10 :	0.336269	50.867308	76.489	14.000 ->	62.489	30 :	23.394726	42.556098	1119.560	794.000 ->	325.560
11 :	21.031561	52.097275	141.447	112.000 ->	29.447	31 :	27.909290	43.192709	38.243	5.000 ->	33.243
12 :	5.809620	52.178407	96.846	53.000 ->	43.846	32 :	21.035229	52.474971	139.882	79.000 ->	60.882
13 :	20.669911	53.892409	187.031	145.000 ->	42.031	33 :	4.358834	50.798668	157.470	101.000 ->	56.470
14 :	16.704471	40.649120	535.657	471.000 ->	64.657	34 :	7.668582	47.567070	504.933	475.000 ->	29.933
15 :	11.646799	44.519965	50.057	10.000 ->	40.057	35 :	7.209528	45.958605	1683.440	1918.000 ->	-234.560
16 :	14.989779	36.876079	126.244	101.000 ->	25.244	36 :	6.102058	46.454101	1258.260	1213.000 ->	45.260
17 :	11.925486	57.395302	45.547	13.000 ->	32.547	37 :	8.670456	46.659734	2094.220	2176.000 ->	-81.780
18 :	19.281520	47.789608	291.748	209.000 ->	82.748	38 :	8.498912	47.713004	563.324	498.000 ->	65.324
19 :	13.066073	52.379286	144.420	30.000 ->	114.420	39 :	8.940319	45.851463	429.707	360.000 ->	69.707
20 :	24.058785	56.948625	34.702	0.000 ->	34.702	40 :	10.100730	46.698581	1612.690	1519.000 ->	93.690
						41 :	14.969324	50.817232	338.631	320.000 ->	18.631

42 :	13.666705	49.462587	547.358	518.000 ->	29.358	85 :	18.060776	42.657912	46.673	21.000 ->	25.673
43 :	17.246431	49.505901	303.702	259.000 ->	44.702	86 :	18.711655	45.153943	146.095	76.000 ->	70.095
44 :	15.935217	49.858445	427.682	396.000 ->	31.682	87 :	16.438445	43.506635	47.627	1.000 ->	46.627
45 :	11.228598	52.335512	153.490	87.000 ->	66.490	88 :	15.978647	45.806199	160.909	120.000 ->	40.909
46 :	9.681841	50.508463	334.877	282.000 ->	52.877	89 :	15.679620	44.845120	713.172	657.000 ->	56.172
47 :	6.761590	50.673819	217.982	168.000 ->	49.982	90 :	21.632365	48.377004	155.772	117.000 ->	38.772
48 :	9.283692	48.400209	760.506	723.000 ->	37.506	91 :	20.670770	46.319570	142.470	91.000 ->	51.470
49 :	7.295758	49.914766	526.493	450.000 ->	76.493	92 :	18.619237	47.255668	234.628	159.000 ->	75.628
50 :	12.487501	50.840896	334.308	283.000 ->	51.308	93 :	-9.583063	51.871472	62.621	72.000 ->	-9.379
51 :	11.141565	48.917419	597.626	533.000 ->	64.626	94 :	-6.541173	53.711583	135.615	68.000 ->	67.615
52 :	8.007257	52.363932	128.934	62.000 ->	66.934	95 :	-7.339016	55.371616	82.705	1.000 ->	81.705
53 :	12.088951	54.177972	49.546	3.000 ->	46.546	96 :	16.867878	41.132069	63.116	2.000 ->	61.116
54 :	23.932619	38.078547	514.553	392.000 ->	122.553	97 :	15.094687	37.501178	51.971	9.000 ->	42.971
55 :	12.499933	55.738882	87.938	28.000 ->	59.938	98 :	11.788148	42.096242	57.488	7.000 ->	50.488
56 :	12.499819	55.738824	86.715	40.000 ->	46.715	99 :	14.850322	40.649523	86.635	28.000 ->	58.635
57 :	27.248784	57.842324	107.504	96.000 ->	11.504	100 :	8.925651	44.412079	49.339	48.000 ->	1.339
58 :	24.380271	59.463565	84.272	29.000 ->	55.272	101 :	11.075102	42.871680	68.406	105.000 ->	-36.594
59 :	-2.459192	36.852473	125.048	137.000 ->	-11.952	102 :	11.231004	43.803827	144.673	56.000 ->	88.673
60 :	2.157415	41.350939	67.660	10.000 ->	57.660	103 :	12.452550	41.922461	201.931	36.000 ->	165.931
61 :	-1.024620	41.721528	269.634	194.000 ->	75.634	104 :	14.213193	42.464939	68.747	2.000 ->	66.747
62 :	-8.398931	43.364369	66.957	1.000 ->	65.957	105 :	13.759409	45.647203	56.199	18.000 ->	38.199
63 :	2.624605	39.552594	59.085	2.000 ->	57.085	106 :	9.115185	39.210538	60.143	13.000 ->	47.143
64 :	-4.110514	38.686741	763.213	700.000 ->	63.213	107 :	5.810079	52.178523	89.310	53.000 ->	36.310
65 :	22.976524	59.822697	24.858	1.000 ->	23.858	108 :	23.371464	55.913575	164.821	136.000 ->	28.821
66 :	-0.561842	44.815672	53.915	5.000 ->	48.915	109 :	25.298666	54.653125	240.831	169.000 ->	71.831
67 :	3.735284	46.620143	257.485	236.000 ->	21.485	110 :	24.410065	57.315888	26.406	12.000 ->	14.406
68 :	-4.503849	48.407887	104.411	46.000 ->	58.411	111 :	21.000909	56.514952	36.258	2.000 ->	34.258
69 :	0.105997	49.481896	53.632	65035.000 ->	-64981.368	112 :	21.851980	57.554411	40.611	6.000 ->	34.611
70 :	6.128414	48.762776	241.724	194.000 ->	47.724	113 :	-4.249170	40.427214	815.088	831.000 ->	-15.912
71 :	2.424987	48.844424	126.148	58.000 ->	68.148	114 :	22.863652	42.001531	1264.160	1111.000 ->	153.160
72 :	-0.199389	46.990445	133.433	87.000 ->	46.433	115 :	11.441623	59.125635	141.486	90.000 ->	51.486
73 :	-1.685527	55.212790	144.401	79.000 ->	65.401	116 :	11.925486	57.395302	45.558	13.000 ->	32.558
74 :	-2.782971	56.478513	57.779	1.000 ->	56.779	117 :	9.784687	47.515331	1090.310	825.000 ->	265.310
75 :	-1.450672	50.930791	98.586	10.000 ->	88.586	118 :	19.201822	50.512247	367.231	300.000 ->	67.231
76 :	-2.386369	54.446576	356.112	304.000 ->	52.112	119 :	15.552539	52.578390	101.438	38.000 ->	63.438
77 :	-1.192268	52.940670	98.465	32.000 ->	66.465	120 :	22.359563	53.734044	154.602	120.000 ->	34.602
78 :	25.566236	40.928037	182.585	126.000 ->	56.585	121 :	18.326514	54.827479	70.796	65035.000 ->	-64964.204
79 :	20.664811	39.734223	598.607	571.000 ->	27.607	122 :	16.853840	54.587695	33.861	65035.000 ->	-65001.139
80 :	21.320333	37.644130	26.885	1.000 ->	25.885	123 :	-6.941155	41.022116	221.752	144.000 ->	77.752
81 :	15.493466	47.067165	540.068	430.000 ->	110.068	124 :	-7.051678	38.878854	229.935	166.000 ->	63.935
82 :	0.335639	50.867421	70.809	14.000 ->	56.809	125 :	-8.707011	41.196354	70.070	29.000 ->	41.070
83 :	14.585901	45.255471	182.398	163.000 ->	19.398	126 :	-8.668737	37.100047	55.347	35.000 ->	20.347
84 :	15.570243	45.578563	268.988	185.000 ->	83.988	127 :	27.204036	47.242718	222.944	67.000 ->	155.944

128 :	21.345945	45.738362	140.156	95.000 ->	45.156	137 :	18.884517	49.165908	415.054	507.000 ->	-91.946
129 :	28.624054	44.588148	156.390	106.000 ->	50.390	138 :	1.480752	43.560778	207.106	145.000 ->	62.106
130 :	11.217940	58.353462	45.196	65035.000 ->	-64989.804	139 :	26.717480	38.426617	58.650	10.000 ->	48.650
131 :	18.090914	59.322331	35.084	65035.000 ->	-64999.916	140 :	22.452611	48.562870	273.801	201.000 ->	72.801
132 :	15.185443	46.286396	342.172	316.000 ->	26.172	141 :	18.367308	57.653880	79.788	31.000 ->	48.788
133 :	16.476891	46.565884	385.191	281.000 ->	104.191	142 :	6.604485	52.914599	82.275	17.000 ->	65.275
134 :	13.643386	45.503793	323.127	124.000 ->	199.127	143 :	12.878487	49.143966	660.272	587.000 ->	73.272
135 :	18.729417	47.836647	261.732	232.000 ->	29.732	144 :	12.878945	49.144253	665.937	587.000 ->	78.937
136 :	20.323501	49.034587	743.178	689.000 ->	54.178						

F.2 Differences between GPS points and GETASSE30 DEM – Nearest neighbour

DEM "../DEM_GETASSE30/EUROPA_WGS84" controlled by "EUVN_ETRF.BLH" GCPs:

NUMBER OF GCPs IN INPUT	217
NUMBER OF PROCESSED GCPs	169 (77.9%)
NUMBER OF GCPs OUTSITE THE DEM	48 (22.1%)
ELEVATION DIFFERENCES :	
. ARITHMETIC MEAN	10.828
. QUADRATIC MEAN	32.368
. STANDARD DEVIATION	30.503
. MINIMUM	-127.560
. MAXIMUM	100.931

GCP #	EASTING	NORTHING	GCP ELEVAT.	DEM ELEVAT.	DIFFERENCE	15 :	11.646799	44.519965	50.057	49.000 ->	1.057
1 :	17.073455	52.276957	124.366	111.000 ->	13.366	16 :	14.989779	36.876079	126.244	152.000 ->	-25.756
2 :	4.359235	50.797807	149.668	134.000 ->	15.668	17 :	11.925486	57.395302	45.547	36.000 ->	9.547
3 :	8.972749	39.135881	238.378	162.000 ->	76.378	18 :	19.281520	47.789608	291.748	299.000 ->	-7.252
4 :	3.399645	50.933714	63.894	55.000 ->	8.894	19 :	13.066073	52.379286	144.420	105.000 ->	39.420
5 :	4.594950	50.094902	282.695	217.000 ->	65.695	20 :	24.058785	56.948625	34.702	33.000 ->	1.702
6 :	0.492343	40.820895	107.810	58.000 ->	49.810	21 :	-6.205649	36.464351	84.180	66.000 ->	18.180
7 :	14.785635	49.913677	592.594	555.000 ->	37.594	22 :	11.877930	45.406732	84.043	63.000 ->	21.043
8 :	6.920585	43.754725	1319.310	1303.000 ->	16.310	23 :	-3.951976	40.443603	647.362	668.000 ->	-20.638
9 :	15.493466	47.067107	538.290	472.000 ->	66.290	24 :	18.367308	57.653880	79.787	80.000 ->	-0.213
10 :	0.336269	50.867308	76.489	52.000 ->	24.489	25 :	12.878888	49.144195	666.026	627.000 ->	39.026
11 :	21.031561	52.097275	141.447	135.000 ->	6.447	26 :	7.465296	46.877115	956.342	908.000 ->	48.342
12 :	5.809620	52.178407	96.846	82.000 ->	14.846	27 :	15.598375	48.654545	457.606	440.000 ->	17.606
13 :	20.669911	53.892409	187.031	180.000 ->	7.031	28 :	13.683435	46.554197	687.372	677.000 ->	10.372
14 :	16.704471	40.649120	535.657	500.000 ->	35.657	29 :	12.082992	47.496026	630.059	569.000 ->	61.059

30 :	2.927413	51.234919	54.461	48.000 ->	6.461	73 :	22.976524	59.822697	24.858	29.000 ->	-4.142
31 :	27.482093	42.483503	41.743	38.000 ->	3.743	74 :	8.762702	41.927389	96.797	48.000 ->	48.797
32 :	23.394726	42.556098	1119.560	1029.000 ->	90.560	75 :	-0.561842	44.815672	53.915	53.000 ->	0.915
33 :	27.909290	43.192709	38.243	37.000 ->	1.243	76 :	3.735284	46.620143	257.485	258.000 ->	-0.515
34 :	21.035229	52.474971	139.882	131.000 ->	8.882	77 :	-4.503849	48.407887	104.411	97.000 ->	7.411
35 :	4.358834	50.798668	157.470	134.000 ->	23.470	78 :	0.105997	49.481896	53.632	49.000 ->	4.632
36 :	7.668582	47.567070	504.933	480.000 ->	24.933	79 :	5.353775	43.278770	61.797	49.000 ->	12.797
37 :	7.209528	45.958605	1683.440	1811.000 ->	-127.560	80 :	6.128414	48.762776	241.724	246.000 ->	-4.276
38 :	6.102058	46.454101	1258.260	1363.000 ->	-104.740	81 :	2.424987	48.844424	126.148	94.000 ->	32.148
39 :	8.670456	46.659734	2094.220	2214.000 ->	-119.780	82 :	-1.681746	43.395136	54.264	80.000 ->	-25.736
40 :	8.498912	47.713004	563.324	541.000 ->	22.324	83 :	-0.199389	46.990445	133.433	133.000 ->	0.433
41 :	8.940319	45.851463	429.707	406.000 ->	23.707	84 :	-1.685527	55.212790	144.401	136.000 ->	8.401
42 :	10.100730	46.698581	1612.690	1578.000 ->	34.690	85 :	-5.924040	54.622245	67.897	55.000 ->	12.897
43 :	14.969324	50.817232	338.631	360.000 ->	-21.369	86 :	-2.782971	56.478513	57.779	51.000 ->	6.779
44 :	13.666705	49.462587	547.358	599.000 ->	-51.642	87 :	-5.355838	36.131807	45.469	42.000 ->	3.469
45 :	17.246431	49.505901	303.702	284.000 ->	19.702	88 :	-1.450672	50.930791	98.586	67.000 ->	31.586
46 :	15.935217	49.858445	427.682	424.000 ->	3.682	89 :	-2.386369	54.446576	356.112	353.000 ->	3.112
47 :	11.228598	52.335512	153.490	146.000 ->	7.490	90 :	-5.542908	50.102924	57.507	53.000 ->	4.507
48 :	9.681841	50.508463	334.877	330.000 ->	4.877	91 :	-1.192268	52.940670	98.465	77.000 ->	21.465
49 :	8.717037	53.868059	45.021	40.000 ->	5.021	92 :	25.566236	40.928037	182.585	246.000 ->	-63.415
50 :	6.761590	50.673819	217.982	217.000 ->	0.982	93 :	20.664811	39.734223	598.607	566.000 ->	32.607
51 :	9.283692	48.400209	760.506	743.000 ->	17.506	94 :	21.320333	37.644130	26.885	23.000 ->	3.885
52 :	7.295758	49.914766	526.493	509.000 ->	17.493	95 :	15.493466	47.067165	540.068	472.000 ->	68.068
53 :	12.487501	50.840896	334.308	327.000 ->	7.308	96 :	0.335639	50.867421	70.809	52.000 ->	18.809
54 :	11.141565	48.917419	597.626	586.000 ->	11.626	97 :	14.585901	45.255471	182.398	168.000 ->	14.398
55 :	8.007257	52.363932	128.934	121.000 ->	7.934	98 :	15.570243	45.578563	268.988	199.000 ->	69.988
56 :	12.088951	54.177972	49.546	42.000 ->	7.546	99 :	18.060776	42.657912	46.673	67.000 ->	-20.327
57 :	23.932619	38.078547	514.553	519.000 ->	-4.447	100 :	18.711655	45.153943	146.095	128.000 ->	18.095
58 :	12.499933	55.738882	87.938	81.000 ->	6.938	101 :	16.438445	43.506635	47.627	56.000 ->	-8.373
59 :	9.962361	57.595037	42.208	53.000 ->	-10.792	102 :	15.978647	45.806199	160.909	164.000 ->	-3.091
60 :	8.439783	55.460139	43.303	40.000 ->	3.303	103 :	13.629348	45.084043	53.589	53.000 ->	0.589
61 :	11.924512	54.572108	40.619	38.000 ->	2.619	104 :	15.679620	44.845120	713.172	758.000 ->	-44.828
62 :	12.499819	55.738824	86.715	81.000 ->	5.715	105 :	21.632365	48.377004	155.772	135.000 ->	20.772
63 :	27.248784	57.842324	107.504	110.000 ->	-2.496	106 :	20.670770	46.319570	142.470	128.000 ->	14.470
64 :	24.380271	59.463565	84.272	51.000 ->	33.272	107 :	18.619237	47.255668	234.628	228.000 ->	6.628
65 :	-0.481227	38.338897	60.347	49.000 ->	11.347	108 :	-9.583063	51.871472	62.621	73.000 ->	-10.379
66 :	-2.459192	36.852473	125.048	87.000 ->	38.048	109 :	-6.541173	53.711583	135.615	94.000 ->	41.615
67 :	2.157415	41.350939	67.660	49.000 ->	18.660	110 :	-7.339016	55.371616	82.705	67.000 ->	15.705
68 :	-1.024620	41.721528	269.634	258.000 ->	11.634	111 :	16.867878	41.132069	63.116	47.000 ->	16.116
69 :	-8.398931	43.364369	66.957	61.000 ->	5.957	112 :	15.094687	37.501178	51.971	50.000 ->	1.971
70 :	2.624605	39.552594	59.085	49.000 ->	10.085	113 :	11.788148	42.096242	57.488	70.000 ->	-12.512
71 :	-4.110514	38.686741	763.213	732.000 ->	31.213	114 :	14.850322	40.649523	86.635	77.000 ->	9.635
72 :	-3.789428	43.461426	59.281	50.000 ->	9.281	115 :	8.925651	44.412079	49.339	58.000 ->	-8.661

116 :	11.075102	42.871680	68.406	111.000 ->	-42.594	143 :	-6.941155	41.022116	221.752	229.000 ->	-7.248
117 :	11.231004	43.803827	144.673	98.000 ->	46.673	144 :	-9.426130	38.690064	65.808	53.000 ->	12.808
118 :	12.452550	41.922461	201.931	101.000 ->	100.931	145 :	-7.051678	38.878854	229.935	222.000 ->	7.935
119 :	14.213193	42.464939	68.747	48.000 ->	20.747	146 :	-8.707011	41.196354	70.070	73.000 ->	-2.930
120 :	13.759409	45.647203	56.199	68.000 ->	-11.801	147 :	-8.668737	37.100047	55.347	52.000 ->	3.347
121 :	9.115185	39.210538	60.143	50.000 ->	10.143	148 :	27.204036	47.242718	222.944	177.000 ->	45.944
122 :	5.810079	52.178523	89.310	82.000 ->	7.310	149 :	28.659006	44.168573	37.216	33.000 ->	4.216
123 :	23.371464	55.913575	164.821	157.000 ->	7.821	150 :	21.345945	45.738362	140.156	132.000 ->	8.156
124 :	25.298666	54.653125	240.831	222.000 ->	18.831	151 :	28.624054	44.588148	156.390	143.000 ->	13.390
125 :	21.083013	55.729771	29.329	24.000 ->	5.329	152 :	15.589036	56.104256	35.189	32.000 ->	3.189
126 :	24.410065	57.315888	26.406	28.000 ->	-1.594	153 :	11.217940	58.353462	45.196	36.000 ->	9.196
127 :	21.000909	56.514952	36.258	28.000 ->	8.258	154 :	18.090914	59.322331	35.084	30.000 ->	5.084
128 :	21.537941	57.395876	27.941	31.000 ->	-3.059	155 :	15.185443	46.286396	342.172	321.000 ->	21.172
129 :	21.851980	57.554411	40.611	37.000 ->	3.611	156 :	16.476891	46.565884	385.191	316.000 ->	69.191
130 :	-4.249170	40.427214	815.088	843.000 ->	-27.912	157 :	13.643386	45.503793	323.127	261.000 ->	62.127
131 :	22.863652	42.001531	1264.160	1203.000 ->	61.160	158 :	18.729417	47.836647	261.732	185.000 ->	76.732
132 :	7.554792	58.006474	43.822	40.000 ->	3.822	159 :	20.323501	49.034587	743.178	688.000 ->	55.178
133 :	5.768998	58.961827	45.534	43.000 ->	2.534	160 :	18.884517	49.165908	415.054	480.000 ->	-64.946
134 :	11.441623	59.125635	141.486	147.000 ->	-5.514	161 :	5.219359	53.362711	56.091	41.000 ->	15.091
135 :	11.925486	57.395302	45.558	36.000 ->	9.558	162 :	1.480752	43.560778	207.106	194.000 ->	13.106
136 :	9.784687	47.515331	1090.310	1029.000 ->	61.310	163 :	27.845061	40.390488	40.005	38.000 ->	2.005
137 :	19.201822	50.512247	367.231	370.000 ->	-2.769	164 :	26.717480	38.426617	58.650	80.000 ->	-21.350
138 :	15.552539	52.578390	101.438	99.000 ->	2.438	165 :	22.452611	48.562870	273.801	237.000 ->	36.801
139 :	22.359563	53.734044	154.602	153.000 ->	1.602	166 :	18.367308	57.653880	79.788	80.000 ->	-0.212
140 :	18.326514	54.827479	70.796	70.000 ->	0.796	167 :	6.604485	52.914599	82.275	65.000 ->	17.275
141 :	14.226771	53.927591	42.205	44.000 ->	-1.795	168 :	12.878487	49.143966	660.272	627.000 ->	33.272
142 :	16.853840	54.587695	33.861	32.000 ->	1.861	169 :	12.878945	49.144253	665.937	627.000 ->	38.937

F.3 Differences between GPS points and GETASSE30 DEM – Bi-linear

```
DEM "../DEM_GETASSE30/EUROPA_WGS84" controlled by "EUVN_ETRF.BLH" GCPs:
NUMBER OF GCPs IN INPUT          :      217
NUMBER OF PROCESSED GCPs         :      169 (77.9%)
NUMBER OF GCPs OUTSITE THE DEM   :       48 (22.1%)
ELEVATION DIFFERENCES :
. ARITHMETIC MEAN                :       11.304
. QUADRATIC  MEAN                :       34.017
. STANDARD DEVIATION             :       32.083
. MINIMUM                        :     -181.091
```

. MAXIMUM : 125.763

GCP #	EASTING	NORTHING	GCP ELEVAT.	DEM ELEVAT.	DIFFERENCE
1 :	17.073455	52.276957	124.366	112.812 ->	11.554
2 :	4.359235	50.797807	149.668	133.450 ->	16.218
3 :	8.972749	39.135881	238.378	167.832 ->	70.546
4 :	3.399645	50.933714	63.894	54.994 ->	8.900
5 :	4.594950	50.094902	282.695	243.019 ->	39.676
6 :	0.492343	40.820895	107.810	64.683 ->	43.127
7 :	14.785635	49.913677	592.594	529.154 ->	63.440
8 :	6.920585	43.754725	1319.310	1241.729 ->	77.581
9 :	15.493466	47.067107	538.290	479.032 ->	59.258
10 :	0.336269	50.867308	76.489	55.476 ->	21.013
11 :	21.031561	52.097275	141.447	136.241 ->	5.206
12 :	5.809620	52.178407	96.846	83.807 ->	13.039
13 :	20.669911	53.892409	187.031	179.426 ->	7.605
14 :	16.704471	40.649120	535.657	504.194 ->	31.463
15 :	11.646799	44.519965	50.057	48.379 ->	1.678
16 :	14.989779	36.876079	126.244	147.778 ->	-21.534
17 :	11.925486	57.395302	45.547	39.760 ->	5.787
18 :	19.281520	47.789608	291.748	307.604 ->	-15.856
19 :	13.066073	52.379286	144.420	113.480 ->	30.940
20 :	24.058785	56.948625	34.702	33.045 ->	1.657
21 :	-6.205649	36.464351	84.180	60.023 ->	24.157
22 :	11.877930	45.406732	84.043	60.589 ->	23.454
23 :	-3.951976	40.443603	647.362	664.634 ->	-17.272
24 :	18.367308	57.653880	79.787	78.403 ->	1.384
25 :	12.878888	49.144195	666.026	627.837 ->	38.189
26 :	7.465296	46.877115	956.342	909.589 ->	46.753
27 :	15.598375	48.654545	457.606	435.192 ->	22.414
28 :	13.683435	46.554197	687.372	654.312 ->	33.060
29 :	12.082992	47.496026	630.059	575.422 ->	54.637
30 :	2.927413	51.234919	54.461	47.669 ->	6.792
31 :	27.482093	42.483503	41.743	38.000 ->	3.743
32 :	23.394726	42.556098	1119.560	993.797 ->	125.763
33 :	27.909290	43.192709	38.243	37.494 ->	0.749
34 :	21.035229	52.474971	139.882	130.440 ->	9.442
35 :	4.358834	50.798668	157.470	133.594 ->	23.876
36 :	7.668582	47.567070	504.933	482.512 ->	22.421
37 :	7.209528	45.958605	1683.440	1864.531 ->	-181.091
38 :	6.102058	46.454101	1258.260	1317.040 ->	-58.780
39 :	8.670456	46.659734	2094.220	2153.840 ->	-59.620
40 :	8.498912	47.713004	563.324	527.537 ->	35.787
41 :	8.940319	45.851463	429.707	403.264 ->	26.443
42 :	10.100730	46.698581	1612.690	1634.494 ->	-21.804
43 :	14.969324	50.817232	338.631	361.892 ->	-23.261
44 :	13.666705	49.462587	547.358	571.505 ->	-24.147
45 :	17.246431	49.505901	303.702	280.865 ->	22.837
46 :	15.935217	49.858445	427.682	424.213 ->	3.469
47 :	11.228598	52.335512	153.490	143.051 ->	10.439
48 :	9.681841	50.508463	334.877	328.023 ->	6.854
49 :	8.717037	53.868059	45.021	39.806 ->	5.215
50 :	6.761590	50.673819	217.982	216.771 ->	1.211
51 :	9.283692	48.400209	760.506	743.031 ->	17.475
52 :	7.295758	49.914766	526.493	498.273 ->	28.220
53 :	12.487501	50.840896	334.308	331.205 ->	3.103
54 :	11.141565	48.917419	597.626	586.354 ->	11.272
55 :	8.007257	52.363932	128.934	121.713 ->	7.221
56 :	12.088951	54.177972	49.546	39.988 ->	9.558
57 :	23.932619	38.078547	514.553	501.339 ->	13.214
58 :	12.499933	55.738882	87.938	77.902 ->	10.036
59 :	9.962361	57.595037	42.208	45.916 ->	-3.708
60 :	8.439783	55.460139	43.303	40.000 ->	3.303
61 :	11.924512	54.572108	40.619	38.000 ->	2.619
62 :	12.499819	55.738824	86.715	77.843 ->	8.872
63 :	27.248784	57.842324	107.504	109.946 ->	-2.442
64 :	24.380271	59.463565	84.272	54.129 ->	30.143
65 :	-0.481227	38.338897	60.347	49.000 ->	11.347
66 :	-2.459192	36.852473	125.048	96.173 ->	28.875
67 :	2.157415	41.350939	67.660	50.107 ->	17.553
68 :	-1.024620	41.721528	269.634	260.437 ->	9.197
69 :	-8.398931	43.364369	66.957	66.484 ->	0.473
70 :	2.624605	39.552594	59.085	49.587 ->	9.498
71 :	-4.110514	38.686741	763.213	744.151 ->	19.062
72 :	-3.789428	43.461426	59.281	50.000 ->	9.281
73 :	22.976524	59.822697	24.858	24.827 ->	0.031
74 :	8.762702	41.927389	96.797	48.000 ->	48.797
75 :	-0.561842	44.815672	53.915	54.821 ->	-0.906
76 :	3.735284	46.620143	257.485	263.942 ->	-6.457
77 :	-4.503849	48.407887	104.411	96.610 ->	7.801
78 :	0.105997	49.481896	53.632	47.181 ->	6.451
79 :	5.353775	43.278770	61.797	60.163 ->	1.634
80 :	6.128414	48.762776	241.724	270.605 ->	-28.881
81 :	2.424987	48.844424	126.148	94.967 ->	31.181

82 :	-1.681746	43.395136	54.264	66.099 ->	-11.835	125 :	21.083013	55.729771	29.329	22.736 ->	6.593
83 :	-0.199389	46.990445	133.433	133.753 ->	-0.320	126 :	24.410065	57.315888	26.406	28.816 ->	-2.410
84 :	-1.685527	55.212790	144.401	136.748 ->	7.653	127 :	21.000909	56.514952	36.258	28.494 ->	7.764
85 :	-5.924040	54.622245	67.897	55.000 ->	12.897	128 :	21.537941	57.395876	27.941	26.148 ->	1.793
86 :	-2.782971	56.478513	57.779	53.485 ->	4.294	129 :	21.851980	57.554411	40.611	34.225 ->	6.386
87 :	-5.355838	36.131807	45.469	100.590 ->	-55.121	130 :	-4.249170	40.427214	815.088	846.380 ->	-31.292
88 :	-1.450672	50.930791	98.586	63.803 ->	34.783	131 :	22.863652	42.001531	1264.160	1184.369 ->	79.791
89 :	-2.386369	54.446576	356.112	371.459 ->	-15.347	132 :	7.554792	58.006474	43.822	40.000 ->	3.822
90 :	-5.542908	50.102924	57.507	59.768 ->	-2.261	133 :	5.768998	58.961827	45.534	43.000 ->	2.534
91 :	-1.192268	52.940670	98.465	78.145 ->	20.320	134 :	11.441623	59.125635	141.486	145.177 ->	-3.691
92 :	25.566236	40.928037	182.585	220.174 ->	-37.589	135 :	11.925486	57.395302	45.558	39.760 ->	5.798
93 :	20.664811	39.734223	598.607	571.592 ->	27.015	136 :	9.784687	47.515331	1090.310	995.169 ->	95.141
94 :	21.320333	37.644130	26.885	23.635 ->	3.250	137 :	19.201822	50.512247	367.231	370.119 ->	-2.888
95 :	15.493466	47.067165	540.068	479.125 ->	60.943	138 :	15.552539	52.578390	101.438	99.488 ->	1.950
96 :	0.335639	50.867421	70.809	55.130 ->	15.679	139 :	22.359563	53.734044	154.602	151.982 ->	2.621
97 :	14.585901	45.255471	182.398	143.241 ->	39.157	140 :	18.326514	54.827479	70.796	68.668 ->	2.128
98 :	15.570243	45.578563	268.988	209.594 ->	59.394	141 :	14.226771	53.927591	42.205	41.296 ->	0.909
99 :	18.060776	42.657912	46.673	72.476 ->	-25.803	142 :	16.853840	54.587695	33.861	33.459 ->	0.402
100 :	18.711655	45.153943	146.095	129.914 ->	16.181	143 :	-6.941155	41.022116	221.752	234.550 ->	-12.798
101 :	16.438445	43.506635	47.627	51.167 ->	-3.540	144 :	-9.426130	38.690064	65.808	53.000 ->	12.808
102 :	15.978647	45.806199	160.909	161.905 ->	-0.996	145 :	-7.051678	38.878854	229.935	223.962 ->	5.973
103 :	13.629348	45.084043	53.589	48.099 ->	5.490	146 :	-8.707011	41.196354	70.070	65.635 ->	4.435
104 :	15.679620	44.845120	713.172	742.703 ->	-29.531	147 :	-8.668737	37.100047	55.347	58.280 ->	-2.933
105 :	21.632365	48.377004	155.772	143.887 ->	11.885	148 :	27.204036	47.242718	222.944	170.510 ->	52.434
106 :	20.670770	46.319570	142.470	128.664 ->	13.806	149 :	28.659006	44.168573	37.216	33.000 ->	4.216
107 :	18.619237	47.255668	234.628	213.072 ->	21.556	150 :	21.345945	45.738362	140.156	131.592 ->	8.564
108 :	-9.583063	51.871472	62.621	110.272 ->	-47.651	151 :	28.624054	44.588148	156.390	140.789 ->	15.601
109 :	-6.541173	53.711583	135.615	112.329 ->	23.286	152 :	15.589036	56.104256	35.189	32.000 ->	3.189
110 :	-7.339016	55.371616	82.705	69.613 ->	13.092	153 :	11.217940	58.353462	45.196	39.091 ->	6.105
111 :	16.867878	41.132069	63.116	47.181 ->	15.935	154 :	18.090914	59.322331	35.084	35.917 ->	-0.833
112 :	15.094687	37.501178	51.971	47.658 ->	4.313	155 :	15.185443	46.286396	342.172	329.825 ->	12.347
113 :	11.788148	42.096242	57.488	56.876 ->	0.612	156 :	16.476891	46.565884	385.191	308.600 ->	76.591
114 :	14.850322	40.649523	86.635	76.209 ->	10.426	157 :	13.643386	45.503793	323.127	253.943 ->	69.184
115 :	8.925651	44.412079	49.339	67.432 ->	-18.093	158 :	18.729417	47.836647	261.732	196.757 ->	64.975
116 :	11.075102	42.871680	68.406	106.220 ->	-37.814	159 :	20.323501	49.034587	743.178	695.092 ->	48.086
117 :	11.231004	43.803827	144.673	100.100 ->	44.573	160 :	18.884517	49.165908	415.054	491.704 ->	-76.650
118 :	12.452550	41.922461	201.931	87.465 ->	114.466	161 :	5.219359	53.362711	56.091	41.000 ->	15.091
119 :	14.213193	42.464939	68.747	49.105 ->	19.642	162 :	1.480752	43.560778	207.106	195.826 ->	11.280
120 :	13.759409	45.647203	56.199	66.645 ->	-10.446	163 :	27.845061	40.390488	40.005	38.000 ->	2.005
121 :	9.115185	39.210538	60.143	51.356 ->	8.787	164 :	26.717480	38.426617	58.650	71.158 ->	-12.508
122 :	5.810079	52.178523	89.310	84.083 ->	5.227	165 :	22.452611	48.562870	273.801	218.340 ->	55.461
123 :	23.371464	55.913575	164.821	155.374 ->	9.447	166 :	18.367308	57.653880	79.788	78.403 ->	1.385
124 :	25.298666	54.653125	240.831	224.789 ->	16.042	167 :	6.604485	52.914599	82.275	65.656 ->	16.619

```
168 :     12.878487    49.143966       660.272       627.916 ->    32.356      169 :     12.878945    49.144253       665.937       627.881 ->    38.056
```

F.4 Differences between GPS points and GETASSE30 DEM – Bi-cubic

```
DEM "../DEM_GETASSE30/EUROPA_WGS84" controlled by "EUVN_ETRF.BLH" GCPs:
NUMBER OF GCPs IN INPUT          :      217
NUMBER OF PROCESSED GCPs         :      169 (77.9%)
NUMBER OF GCPs OUTSITE THE DEM   :       48 (22.1%)
ELEVATION DIFFERENCES :
. ARITHMETIC MEAN                :       12.008
. QUADRATIC  MEAN                :       32.915
. STANDARD DEVIATION             :       30.646
. MINIMUM                        :     -172.570
. MAXIMUM                        :      127.387
```

GCP #	EASTING	NORTHING	GCP ELEVAT.	DEM ELEVAT.	DIFFERENCE		EASTING	NORTHING	GCP ELEVAT.	DEM ELEVAT.	DIFFERENCE
1 :	17.073455	52.276957	124.366	112.723 ->	11.643	23 :	-3.951976	40.443603	647.362	662.525 ->	-15.163
2 :	4.359235	50.797807	149.668	132.284 ->	17.384	24 :	18.367308	57.653880	79.787	78.621 ->	1.166
3 :	8.972749	39.135881	238.378	166.796 ->	71.582	25 :	12.878888	49.144195	666.026	633.368 ->	32.658
4 :	3.399645	50.933714	63.894	55.095 ->	8.799	26 :	7.465296	46.877115	956.342	913.803 ->	42.539
5 :	4.594950	50.094902	282.695	237.366 ->	45.329	27 :	15.598375	48.654545	457.606	441.081 ->	16.525
6 :	0.492343	40.820895	107.810	63.524 ->	44.286	28 :	13.683435	46.554197	687.372	650.284 ->	37.088
7 :	14.785635	49.913677	592.594	533.833 ->	58.761	29 :	12.082992	47.496026	630.059	557.365 ->	72.694
8 :	6.920585	43.754725	1319.310	1245.788 ->	73.522	30 :	2.927413	51.234919	54.461	47.558 ->	6.903
9 :	15.493466	47.067107	538.290	479.585 ->	58.705	31 :	27.482093	42.483503	41.743	37.998 ->	3.745
10 :	0.336269	50.867308	76.489	54.938 ->	21.551	32 :	23.394726	42.556098	1119.560	992.173 ->	127.387
11 :	21.031561	52.097275	141.447	136.975 ->	4.472	33 :	27.909290	43.192709	38.243	37.181 ->	1.062
12 :	5.809620	52.178407	96.846	82.618 ->	14.228	34 :	21.035229	52.474971	139.882	130.629 ->	9.253
13 :	20.669911	53.892409	187.031	179.789 ->	7.242	35 :	4.358834	50.798668	157.470	132.994 ->	24.476
14 :	16.704471	40.649120	535.657	504.578 ->	31.079	36 :	7.668582	47.567070	504.933	485.308 ->	19.625
15 :	11.646799	44.519965	50.057	48.275 ->	1.782	37 :	7.209528	45.958605	1683.440	1856.010 ->	-172.570
16 :	14.989779	36.876079	126.244	148.171 ->	-21.927	38 :	6.102058	46.454101	1258.260	1307.717 ->	-49.457
17 :	11.925486	57.395302	45.547	40.469 ->	5.078	39 :	8.670456	46.659734	2094.220	2137.401 ->	-43.181
18 :	19.281520	47.789608	291.748	307.550 ->	-15.802	40 :	8.498912	47.713004	563.324	525.685 ->	37.639
19 :	13.066073	52.379286	144.420	114.260 ->	30.160	41 :	8.940319	45.851463	429.707	398.204 ->	31.503
20 :	24.058785	56.948625	34.702	33.212 ->	1.490	42 :	10.100730	46.698581	1612.690	1596.687 ->	16.003
21 :	-6.205649	36.464351	84.180	62.822 ->	21.358	43 :	14.969324	50.817232	338.631	360.229 ->	-21.598
22 :	11.877930	45.406732	84.043	60.980 ->	23.063	44 :	13.666705	49.462587	547.358	572.080 ->	-24.722
						45 :	17.246431	49.505901	303.702	283.528 ->	20.174

46 :	15.935217	49.858445	427.682	421.547 ->	6.135	89 :	-2.386369	54.446576	356.112	373.204 ->	-17.092
47 :	11.228598	52.335512	153.490	142.074 ->	11.416	90 :	-5.542908	50.102924	57.507	57.362 ->	0.145
48 :	9.681841	50.508463	334.877	328.169 ->	6.708	91 :	-1.192268	52.940670	98.465	77.148 ->	21.317
49 :	8.717037	53.868059	45.021	39.868 ->	5.153	92 :	25.566236	40.928037	182.585	219.439 ->	-36.854
50 :	6.761590	50.673819	217.982	217.091 ->	0.891	93 :	20.664811	39.734223	598.607	568.993 ->	29.614
51 :	9.283692	48.400209	760.506	742.933 ->	17.573	94 :	21.320333	37.644130	26.885	21.683 ->	5.202
52 :	7.295758	49.914766	526.493	500.350 ->	26.142	95 :	15.493466	47.067165	540.068	479.585 ->	60.483
53 :	12.487501	50.840896	334.308	331.907 ->	2.401	96 :	0.335639	50.867421	70.809	54.185 ->	16.624
54 :	11.141565	48.917419	597.626	586.403 ->	11.223	97 :	14.585901	45.255471	182.398	148.040 ->	34.358
55 :	8.007257	52.363932	128.934	119.959 ->	8.975	98 :	15.570243	45.578563	268.988	213.595 ->	55.393
56 :	12.088951	54.177972	49.546	40.534 ->	9.012	99 :	18.060776	42.657912	46.673	73.635 ->	-26.962
57 :	23.932619	38.078547	514.553	495.031 ->	19.522	100 :	18.711655	45.153943	146.095	131.279 ->	14.816
58 :	12.499933	55.738882	87.938	79.685 ->	8.253	101 :	16.438445	43.506635	47.627	49.399 ->	-1.772
59 :	9.962361	57.595037	42.208	46.441 ->	-4.233	102 :	15.978647	45.806199	160.909	161.443 ->	-0.534
60 :	8.439783	55.460139	43.303	39.611 ->	3.692	103 :	13.629348	45.084043	53.589	47.210 ->	6.379
61 :	11.924512	54.572108	40.619	38.075 ->	2.544	104 :	15.679620	44.845120	713.172	736.048 ->	-22.876
62 :	12.499819	55.738824	86.715	79.685 ->	7.030	105 :	21.632365	48.377004	155.772	140.266 ->	15.506
63 :	27.248784	57.842324	107.504	109.971 ->	-2.467	106 :	20.670770	46.319570	142.470	128.596 ->	13.874
64 :	24.380271	59.463565	84.272	55.676 ->	28.596	107 :	18.619237	47.255668	234.628	215.137 ->	19.491
65 :	-0.481227	38.338897	60.347	46.192 ->	14.155	108 :	-9.583063	51.871472	62.621	101.853 ->	-39.232
66 :	-2.459192	36.852473	125.048	94.285 ->	30.763	109 :	-6.541173	53.711583	135.615	108.040 ->	27.575
67 :	2.157415	41.350939	67.660	48.808 ->	18.852	110 :	-7.339016	55.371616	82.705	69.699 ->	13.006
68 :	-1.024620	41.721528	269.634	260.769 ->	8.865	111 :	16.867878	41.132069	63.116	46.953 ->	16.163
69 :	-8.398931	43.364369	66.957	62.806 ->	4.151	112 :	15.094687	37.501178	51.971	47.473 ->	4.498
70 :	2.624605	39.552594	59.085	49.099 ->	9.986	113 :	11.788148	42.096242	57.488	55.702 ->	1.786
71 :	-4.110514	38.686741	763.213	740.724 ->	22.489	114 :	14.850322	40.649523	86.635	76.195 ->	10.440
72 :	-3.789428	43.461426	59.281	48.260 ->	11.021	115 :	8.925651	44.412079	49.339	61.144 ->	-11.805
73 :	22.976524	59.822697	24.858	26.119 ->	-1.261	116 :	11.075102	42.871680	68.406	111.298 ->	-42.892
74 :	8.762702	41.927389	96.797	45.929 ->	50.868	117 :	11.231004	43.803827	144.673	99.837 ->	44.836
75 :	-0.561842	44.815672	53.915	54.871 ->	-0.956	118 :	12.452550	41.922461	201.931	83.302 ->	118.629
76 :	3.735284	46.620143	257.485	263.692 ->	-6.207	119 :	14.213193	42.464939	68.747	48.891 ->	19.856
77 :	-4.503849	48.407887	104.411	97.902 ->	6.509	120 :	13.759409	45.647203	56.199	69.730 ->	-13.531
78 :	0.105997	49.481896	53.632	47.363 ->	6.269	121 :	9.115185	39.210538	60.143	50.483 ->	9.660
79 :	5.353775	43.278770	61.797	56.707 ->	5.090	122 :	5.810079	52.178523	89.310	83.347 ->	5.963
80 :	6.128414	48.762776	241.724	262.970 ->	-21.246	123 :	23.371464	55.913575	164.821	155.424 ->	9.397
81 :	2.424987	48.844424	126.148	94.233 ->	31.915	124 :	25.298666	54.653125	240.831	226.470 ->	14.361
82 :	-1.681746	43.395136	54.264	65.093 ->	-10.829	125 :	21.083013	55.729771	29.329	22.435 ->	6.894
83 :	-0.199389	46.990445	133.433	134.083 ->	-0.650	126 :	24.410065	57.315888	26.406	29.183 ->	-2.777
84 :	-1.685527	55.212790	144.401	138.244 ->	6.157	127 :	21.000909	56.514952	36.258	29.052 ->	7.206
85 :	-5.924040	54.622245	67.897	53.946 ->	13.951	128 :	21.537941	57.395876	27.941	26.503 ->	1.438
86 :	-2.782971	56.478513	57.779	53.560 ->	4.219	129 :	21.851980	57.554411	40.611	34.141 ->	6.470
87 :	-5.355838	36.131807	45.469	88.148 ->	-42.679	130 :	-4.249170	40.427214	815.088	842.403 ->	-27.315
88 :	-1.450672	50.930791	98.586	63.465 ->	35.121	131 :	22.863652	42.001531	1264.160	1196.877 ->	67.283

132 :	7.554792	58.006474	43.822	37.857 ->	5.965	151 :	28.624054	44.588148	156.390	142.725 ->	13.665
133 :	5.768998	58.961827	45.534	42.585 ->	2.949	152 :	15.589036	56.104256	35.189	32.004 ->	3.185
134 :	11.441623	59.125635	141.486	146.211 ->	-4.725	153 :	11.217940	58.353462	45.196	39.917 ->	5.279
135 :	11.925486	57.395302	45.558	40.469 ->	5.088	154 :	18.090914	59.322331	35.084	34.771 ->	0.313
136 :	9.784687	47.515331	1090.310	1008.774 ->	81.536	155 :	15.185443	46.286396	342.172	323.922 ->	18.250
137 :	19.201822	50.512247	367.231	371.068 ->	-3.837	156 :	16.476891	46.565884	385.191	308.686 ->	76.505
138 :	15.552539	52.578390	101.438	98.918 ->	2.520	157 :	13.643386	45.503793	323.127	261.656 ->	61.471
139 :	22.359563	53.734044	154.602	152.159 ->	2.443	158 :	18.729417	47.836647	261.732	191.301 ->	70.431
140 :	18.326514	54.827479	70.796	69.929 ->	0.867	159 :	20.323501	49.034587	743.178	692.453 ->	50.725
141 :	14.226771	53.927591	42.205	41.565 ->	0.640	160 :	18.884517	49.165908	415.054	485.528 ->	-70.474
142 :	16.853840	54.587695	33.861	32.685 ->	1.176	161 :	5.219359	53.362711	56.091	39.798 ->	16.293
143 :	-6.941155	41.022116	221.752	232.939 ->	-11.187	162 :	1.480752	43.560778	207.106	195.360 ->	11.746
144 :	-9.426130	38.690064	65.808	51.780 ->	14.028	163 :	27.845061	40.390488	40.005	37.030 ->	2.975
145 :	-7.051678	38.878854	229.935	224.384 ->	5.551	164 :	26.717480	38.426617	58.650	71.807 ->	-13.157
146 :	-8.707011	41.196354	70.070	65.521 ->	4.549	165 :	22.452611	48.562870	273.801	223.232 ->	50.569
147 :	-8.668737	37.100047	55.347	56.649 ->	-1.302	166 :	18.367308	57.653880	79.788	78.621 ->	1.167
148 :	27.204036	47.242718	222.944	175.060 ->	47.884	167 :	6.604485	52.914599	82.275	65.809 ->	16.466
149 :	28.659006	44.168573	37.216	32.986 ->	4.230	168 :	12.878487	49.143966	660.272	633.368 ->	26.904
150 :	21.345945	45.738362	140.156	131.726 ->	8.430	169 :	12.878945	49.144253	665.937	633.368 ->	32.569