KARIOS Image Matching Tool: Software User Manual

Author(s): ***Sébastien Saunier***
Patrice Canonici
EDAP+ Task 2 Optical Team

Approval: ***Léonardo De Laurentis***
EDAP+ Technical Officer

EDAP+.SUM.001 *Issue: 2.0* *October, 27 2025*

TELESPAZIO
a LEONARDO and THALES company

AMENDMENT RECORD SHEET

The Amendment Record Sheet below records the history and issue status of this document.

ISSUE	DATE	REASON
1.0	JUNE 2024	Release of the KARIOS Version 1.0
2.0	OCTOBER 2025	Release of the KARIOS Version 2.1.0 (August 2025) https://github.com/telespazio-tim/karios/blob/develop/CHANGELOG.md

TABLE OF CONTENTS

9.6.2

1. INTRODUCTION

1.1 Scope

This Software User Manual (**SUM**) describes the 'Kanade-Lucas-Tomassi (**KLT**)-based Algorithm for Registration of Images from Observing System' (**KARIOS**) Tool developed in the context of the ESA Early Data Assessment Project (**EDAP+**).

1.2 Reference Documents

The following is a list of reference documents with a direct bearing on the content of this proposal. Where referenced in the text, these are identified as [RD-n], where 'n' is the number in the list below:

RD-1. ESA Earthnet Site, https://earth.esa.int/eogateway/activities/edap/vhr-hr-mr-optical-missions

RD-2. "2D Sub-Pixel Disparity Measurement Using QPEC / Medicis " M. Cournet , A. Giros , L. Dumas , J. M. Delvit , D. Greslou , F. Languille , G. Blanchet , S. May , J. Michel
XXIII ISPRS Congress, 12-19 July 2016, Prague, Czech Republic

RD-3. Description of the KLT https://en.wikipedia.org/wiki/Kanade%E2%80%93Lucas%E2%80%93Tomasi_feature_tracker

RD-4. Saunier, S.; Pflug, B.; Lobos, I.M.; Franch, B.; Louis, J.; De Los Reyes, R.; Debaecker, V.; Cadau, E. G.; Boccia, V.; Gascon, F.; Kocaman, S. Sen2Like: Paving the Way towards Harmonization and Fusion of Optical Data. Remote Sens. 2022, 14, 3855. https://doi.org/10.3390/rs14163855

RD-5. EUMETSAT Site https://www.eumetsat.int/GQA-tool

RD-6. Kocaman Aksakal, S., 2013. Geometric Accuracy Investigations of SEVIRI High Resolution Visible (HRV) Level 1.5 Imagery, Remote Sensing, 5, 2475-2491; doi: 10.3390/rs5052475.

RD-7. Kocaman, S., Neuhaus Ch., Baltsavias E., Schindler K., 2015. Geometric Quality Analysis of AVHRR Orthoimages. Remote Sensing, 7(3), 3293-3319., doi: 10.3390/rs070303293.

RD-8. Kocaman, S., Debaecker, V., Bas, S., Saunier, S., Garcia, K., Just, D., 2022. A Comprehensive Geometric Quality Assessment Approach for MSG SEVIRI Imagery. Advances in Space Research. doi: 10.1016/j.asr.2021.11.018.

RD-9. Shi, J., Tomasi, C., 1994. Good features to track. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition Seattle, WA, USA, 21–23 June, 593-600.

RD-10. Lucas, B. D., & Kanade, T., 1981. An iterative image registration technique with an application to stereo vision. International Joint Conference on Artificial Intelligence, 674–679.

RD-11. Tomasi, C., Kanade, T., 1991. Detection and tracking of point features. Carnegie Mellon University Technical Report CMU-CS-91-132, April.

Glossary

The following acronyms and abbreviations have been used in this Report.

EDAP+	Early Data Assessment Project
GQA	Geometrical Quality Assessment
KARIOS	KLT-based Algorithm for Registration of Images from Observing System
KLT	Kanade-Lucas-Tomassi
NMAS	National Map Accuracy Standard
SUM	Software User Manual
TN	Technical Notes
VHR	Very High Resolution
ZNCC	Zero Normalized Cross Correlation

1.3

DESCRIPTION

Scope of the tool

2.

2.1

The EDAP+ Optical team evaluates the **geometric quality** of input optical products by operating the three following assessment procedures as shown in technical reports available in [RD-1]:

- Multi temporal geolocation,
- Absolute geolocation,
- Band-to-band registration.

From a functional point of view, these procedures include two necessary stages, the **image matching** and the **accuracy analysis.** As indicated in Figure 2-1, the KARIOS s/w solution is a tool developed to perform these stages and produce dedicated geometric quality items.

As input, the KARIOS tool requires two images expressed in the same cartographic grid, with the same pixel spacing and same pixel/line size. It is therefore mandatory to prepare data as part of a dedicated pre-processing, including data access, clipping and re-projection, depending on the assessment procedure.

- The multi temporal geolocation procedure aims at evaluating the geolocation stability over time, and the KARIOS tool is used to compare two images from the same mission.
- The absolute geolocation procedure aims at evaluating the geolocation accuracy of an input image against a well-controlled absolute reference image, considered as raster reference and indicated as “REF” in the Figure 2-1 below.
- The band-to-band registration procedure aims at evaluation of the co-registration between image bands within a product.

It is worth noting that choosing which image band to run the tool on, is a user-defined parameter. Grid comparison is usually performed on a twin of image bands originating from the same, or a similar, spectral channel with both images including the information to be matched. This latter comment does not apply to the band-to-band registration procedure.

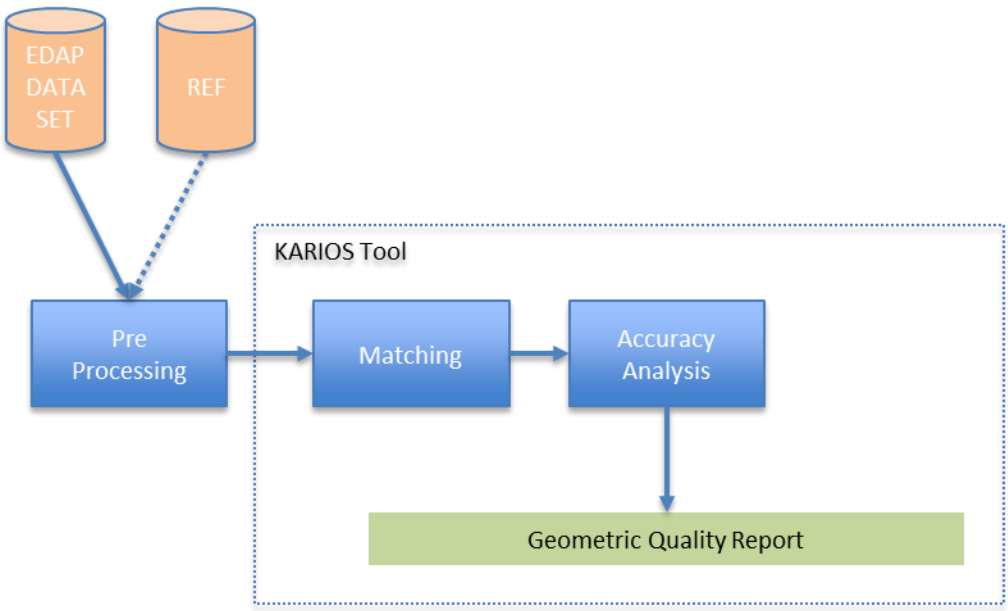


Figure 2-1: Scope of the KARIOS Tool.

The KARIOS tool has been developed with the following high-level characteristics:

- Free and Open-Source licensing policy (GPL)
- Multi Missions (Optical / Radar, Medium Resolution / High Resolution) capabilities
- Python Code
- Quick Processing time
- Reduced Uncertainties

The EDAP+ KARIOS tool enables execution of a geometric processing unitary workflow. The tool proposes **standardised** reports / figures as outputs.

Notably, the report includes planimetric accuracy analysis figures / statistical outputs (displacement maps, histograms, circular error) based on a standardised accuracy metric. In particular, the report is fit for the EDAP+ Technical Notes (**TN**), as shown in the website section dedicated to Very High Resolution (**VHR**) missions [RD-1].

For many reasons, including processing performance, accuracy and handling of images from different types of observing systems (optical, radar), the EDAP+ KARIOS tool does not propose a classic approach for image matching as applied traditionally in the remote sensing domain [RD-2]. The classic approach is based on a zero normalised cross-correlation and algorithms generally include many steps (i.e., dichotomous approach, correlation window, exploration area, similarity measure). The KARIOS algorithm is discussed in the following section.

2.2 Processing Algorithms Overview

2.2.1 Matching

As part of image matching, the proposed image registration approach is based on the KLT feature tracker [RD-3]. This approach is successfully used in Sen2Like [RD-4] and EUMETSAT Geometrical Quality Assessment (**GQA**) projects [RD-5].

A similar method has previously been used for the matching of MSG SEVIRI and AVHRR image series [RD-6], [RD-7], [RD-8]. The KLT is based on methods proposed by Shi and Tomasi [RD-9] and Kanade – Lucas -Tomasi [RD-10], [RD-11].

Image filtering (Laplacian) is applied to each image to enhance the texture . Point selection / matching is undertaken through a motion tracking algorithm (optical flow).

Additionally, statistical outlier detection is applied to eliminate false matches from the results, along with a reverse matching check, to enhance method robustness.

Finally, displacements at the most suitable points in grid are given in both line and pixel directions. This is used as input for accuracy analysis. The number of selected points, so-called keypoints, strongly depend on the configuration specified by the user (configuration is described in Section 6).

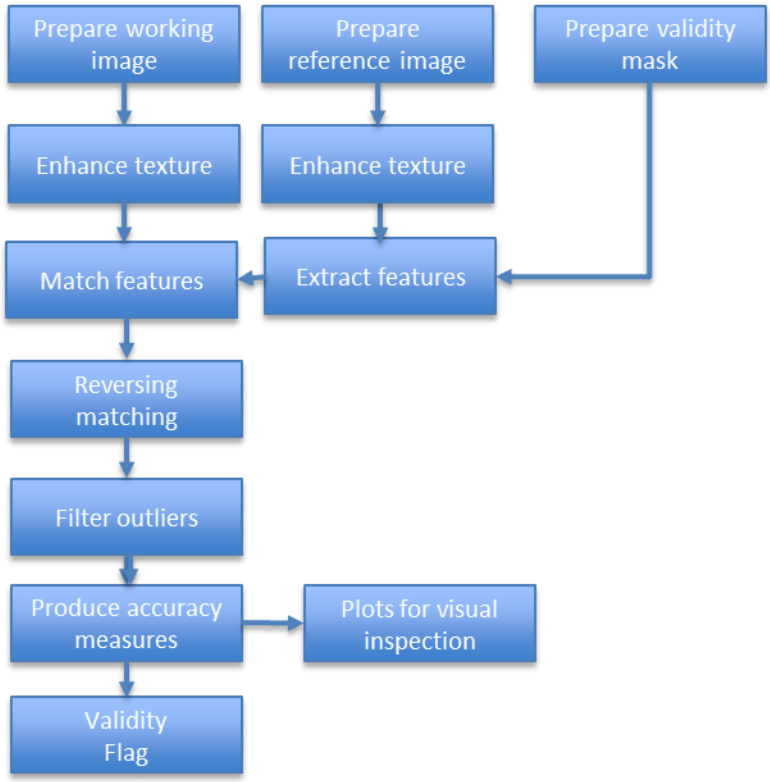


Figure 2-2: KARIOS / KLT Algorithms.

2.2.2 Accuracy Analysis

The KARIOS tool accuracy analysis function performs an analysis of the image matching results. A set of statistics, as a measure of mapping accuracy, is proposed.

Statistics are computed over the KP geometric shift sample. These are expressed in various metrics that are convenient for judging and assessing the data.

For information, the following metrics are computed:

Root Mean Square Error (RMSE)	To describe both random and systematic errors in both directions https://en.wikipedia.org/wiki/Root_mean_square
Minimum / Maximum Error	To report on closest / largest out of all measured location differences
Mean Error	To show the average differences in both directions
Standard Deviation Error	To show the spread of errors in both directions
CE90 / CE95	To analyse the distribution of error in both directions and to check compliance with horizontal accuracy standard as specified by the National Map Accuracy Standard (**NMAS**). https://en.wikipedia.org/wiki/Circular_error_probable

Note that NMAS assumes bivariate normal distributions with zero bias.

Note that circular error report in KARIOS is computed with an empirical approach.

And recalled in the geometric accuracy figures, discussed below:

- Error overview
- Error distribution
- Disparity maps (in pixel direction (dx), in line direction (dy))
- Dependency on terrain relief

2.2.3 Design and Implementation of the tracking method

An implementation of the tracking method is provided under OpenCV and is used for feature point matching. The Candidate point selection is performed using the GoodFeaturesToTrack function and matching with calcOpticalFlowPyrLK.

3. INPUT REQUIREMENTS AND RECOMMENDATIONS

KARIOS requires as input mandatory parameters two images, so called the Monitored image (MON) and the Reference image (REF). The MON image grid is compared against the REF image grid.

It is up to the user to check that both the NOM and REF image grids can be compared. In particular, the same image size, the same geometric sampling, the same geographical area (full overlap) and in some cases the same geographic projection are expected.

KARIOS has not been designed to perform warping or clipping with respect to a user defined region of interest. Instead, through a binary mask file (optional parameter), a region of interest can be defined and matching performed over it.

Furthermore, an additional optional input is the Digital Elevation Model data. As for the binary mask discussed above, the DEM image should be rigorously expressed in the same geometry as the input data and cover the same geographical extent.

The remaining command parameters are discussed in the next section.

KARIOS can be used both as a command-line tool and as a Python library. As command line tool, the structure to be followed by the client is the following one:

```
karios process MONITORED_IMAGE REFERENCE_IMAGE [MASK_FILE] [DEM_FILE] [OPTIONS]
```

Requirements:

- Input images grids should be comparable. The user should take care of data preparation.
- That means geo coded images must have the same footprint, same geotransform information (same cartographic projection, i.e. EPSG code) and same resolution.
- Image pixel resolution should also be square (same X, Y) and unit meter.
- This is also applicable to the DEM and mask files for compatibility requirements.

Recommendation:

- The user shall carefully check the dynamic range of the monitored and reference images, because KARIOS converts these input data into integers.
- For instance, providing float values between 0 and 1 will give very poor results. In that case, it is recommended to multiply the data by 100.
- Input files shall contain only one layer (band) of data, and the format shall be recognized by the GDAL library.

4. COMMAND LINE USAGE AND EXAMPLE

With respect to previous section, the command line structure is

```
karios process MONITORED_IMAGE REFERENCE_IMAGE [MASK_FILE]
[DEM_FILE] [OPTIONS]
```

Where:

- MONITORED_IMAGE: Path to the image to analyse for shifts/changes
- REFERENCE_IMAGE: Path to the stable reference image for comparison
- MASK_FILE: Optional mask file to exclude pixels from matching (use '-' to skip)
- DEM_FILE: Optional DEM file for altitude-based analysis

The content of [OPTIONS] is discussed in the following subsection.

4.1 Command line options

4.1.1 Processing options

Option	Type	Description
--conf	FILE	Configuration file path. Default is the built-in configuration. [default: PWD/karios/configuration/processing_configuration.json]
--resume	Flag	Do not run the KLT matcher, only the accuracy analysis and report generation
--input-pixel-size, -pxs	FLOAT	Input image pixel size in meters. Ignored if the image resolution can be read from the input image

4.1.2 Output options

Option	Type	Description
--out	PATH	Output results folder path [default: results]
--title-prefix, -tp	TEXT	Add a prefix to the title of the generated output charts (limited to 26 characters)
--generate-key-points-mask, -kpm	FLAG	Generate a tiff mask based on KP from KTL
--generate-intermediate-product, -gip	FLAG	Generate a two-band tiff based on KP with band 1 being the X-direction displacement (dx) and band 2 being the Y-direction displacement (dy)
--generate-kp-chips, -gkc	FLAG	Generate chip images of the monitored and reference products centred on the KP

Option	Type	Description
--dem-description	TEXT	DEM source name. Added in output DEM plots (example: "COPERNICUS DEM resampled to 10m")

Advanced options

Option	Type	Description
--enable-large-shift-detection	FLAG	Enable the detection and correction of large pixel shifts

4.1.3

Logging options

4.1.4

Option	Type	Description
--debug, -d	FLAG	Enable Debug mode
--no-log-file	FLAG	Do not log in file (not compatible with --log-file-path)
--log-file-path	PATH	Log file path [default: karios.log]

Examples

4.2

Basic Processing

```
karios process monitored.tif reference.tif
```

With Mask

```
karios process monitored.tif reference.tif mask.tif
```

With Mask and DEM

```
karios process monitored.tif reference.tif mask.tif dem.tif \
  --dem-description "SRTM 30m resample to 10m"
```

DEM Only (No Mask)

```
karios process monitored.tif reference.tif - dem.tif \
  --dem-description "Copernicus DEM 30m"
```

Full Workflow with Options

```
karios process monitored.tif reference.tif mask.tif dem.tif \
  --out ./results \
  --generate-key-points-mask \
  --generate-intermediate-product \
  --title-prefix "MyAnalysis" \
  --dem-description "SRTM 30m" \
  --enable-large-shift-detection
```

Resume Previous Analysis

```
karios process monitored.tif reference.tif mask.tif dem.tif \
  --resume \
  --out ./existing_results
```

LIBRARY USAGE

KARIOS can be used as a Python library in your own applications by providing an API that separates processing configuration from input data.

Basic Example

5.

5.1

```
from karios.api import KariosAPI, RuntimeConfiguration
from karios.core.configuration import ProcessingConfiguration

# Load processing configuration
processing_config = ProcessingConfiguration.from_file("config.json")

# Create runtime configuration (how to process)
runtime_config = RuntimeConfiguration(
    output_directory="./results",
    gen_kp_mask=True,
    gen_delta_raster=True,
    pixel_size=10.0,  # meters
    enable_large_shift_detection=False,
    generate_kp_chips=True,
)

# Initialize API
api = KariosAPI(processing_config, runtime_config)

# Process images (what to process) and generates plots
match_result, accuracy, reports = api.process(
    monitored_image_path="monitored.tif",
    reference_image_path="reference.tif",
    mask_file_path="mask.tif",        # Optional
    dem_file_path="dem.tif"           # Optional
)

# Access results
print(f"CE90: {accuracy.ce90:.3f}")
print(f"Mean shift X: {accuracy.mean_x:.3f} pixels")
print(f"Generated reports: {reports.overview_plot}")
```

5.2

Batch Processing Example

```
# Same configuration, multiple image pairs
image_pairs = [
    ("mon1.tif", "ref1.tif", "mask1.tif", "dem.tif"),
    ("mon2.tif", "ref2.tif", "mask2.tif", "dem.tif"),
    ("mon3.tif", "ref3.tif", None, "dem.tif")  # No mask for this pair
]

results = []
for monitored, reference, mask, dem in image_pairs:
    match, accuracy, reports = api.process(monitored, reference, mask, dem)
    results.append({
```

```
        'pair': (monitored, reference),
        'ce90': accuracy.ce90,
        'mean_shift': (accuracy.mean_x, accuracy.mean_y)
    })
    # ... other statements

    # clean memory
    match = None
    accuracy = None
    reports = None
    monitored = None
    reference = None
    mask = None
    dem = None

    # or eventually use gc.collect()
```

5.3 API Components

- ProcessingConfiguration: KLT parameters, accuracy thresholds, plot settings
- RuntimeConfiguration: Output settings, processing flags, optional descriptions
- KariosAPI: Main processing interface
- MatchResult: Key point matches and image metadata
- AccuracyAnalysis: Statistical metrics (CE90, RMSE, etc.)
- ReportPaths: Generated visualization and product file paths

KARIOS OUTPUT RESULTS

Statistical Files

6. CSV files

6.1

KARIOS generates a CSV file containing all KP detected by the KLT matcher with their displacement measurements and quality metrics.

6.1.1

The CSV file is saved as KLT_matcher_{monitored_filename}_{reference_filename}.csv in the output directory.

Column	Description	Unit	Notes
`x0`	X coordinate of key point in reference image	pixels	Column position (0-based)
`y0`	Y coordinate of key point in reference image	pixels	Row position (0-based)
`dx`	Displacement in X direction (column)	pixels	Positive = eastward shift
`dy`	Displacement in Y direction (row)	pixels	Positive = southward shift
`score`	KLT matching confidence score	0.0-1.0	Higher values indicate better matches
`radial error`	Euclidean distance of displacement	pixels	$\sqrt{(dx^2 + dy^2)}$
`angle`	Direction of displacement	degrees	Measured from east, counter-clockwise
`zncc_score`	Zero-mean Normalized Cross-Correlation score	-1.0 to 1.0	Optional: only if large shift detection is disabled

Figure 6-1: KARIOS CSV File Column.

A text file which indicates the location of KPs in the reference image (x0, Y0), the geometric shift, and the score, as shown in **Figure 6-2**.

```
x0;y0;dx;dy;score;radial error;angle;zncc_score
20.0;3084.0;-1.9419136;-1.1386719;0.19532776;2.2511334;-149.61409;
20.0;3049.0;-1.7128563;-1.0314941;0.16423798;1.9994642;-148.94334;
58.0;3021.0;-1.7995872;-0.8408203;0.4714737;1.9863265;-154.95662;0.6716859690913349
71.0;2896.0;1.0032806;-0.31933594;0.44740295;1.0528758;-17.655842;0.10418937699648298
78.0;2959.0;-0.2112732;-0.7770996;0.3360138;0.8053075;-105.2096;
83.0;2885.0;1.225174;-0.4633789;0.7216797;1.3098745;-20.717358;0.7355966838303492
84.0;2942.0;0.05480194;-0.67871094;0.7667694;0.6809198;-85.38371;0.7676205075531173
85.0;3022.0;-1.465065;-0.8210449;0.75914;1.6794434;-150.73305;0.7425731575300989
89.0;2964.0;-0.25565338;-0.703125;0.82666016;0.74816;-109.98105;0.7597777165307185
91.0;2873.0;0.47808838;-0.71313477;0.5114136;0.8585626;-56.16194;0.7010416219167059
92.0;2854.0;-0.102241516;-0.56152344;0.58839417;0.57075554;-100.31929;0.657369592764875
92.0;2896.0;1.3651276;-0.47558594;0.6948242;1.4455986;-19.207418;0.7706789948244198
98.0;2838.0;0.33000946;-0.62768555;0.58740234;0.7091512;-62.266575;0.6839299565839168
99.0;3081.0;0.5587692;-0.49487305;0.2945099;0.7464063;-41.52965;
101.0;2805.0;0.3591156;-0.42822266;0.596405;0.55887264;-50.016163;0.6756831501802222
102.0;2934.0;0.44795227;-0.5349121;0.79667664;0.697705;-50.05613;0.7139445854372138
105.0;2855.0;-0.08403015;-0.5246582;0.8120117;0.5313448;-99.099304;0.8137731990760797
```

Figure 6-2: KARIOS CSV File example.

In addition, a summary statistic file is produced, 'correl_res.txt'; this is the geolocation accuracy of the product, computed based on sample detailed in the CSV file.

Notes:

- The CSV uses semicolon (;) as separator
- zncc_score is only computed for a selection of key points with KLT score above the confidence threshold
- zncc_score provides additional matching quality assessment using normalized cross-correlation
- All coordinates are in image pixel space (not geographic coordinates)

6.1.2

GeoJSON File

The same type of information is also available in a JSON file for quick insertion into GIS software. The GeoJSON file is more informative, KP information also includes radial error, angle of error, ZNCC score.

```
{
  "type": "FeatureCollection",
  "crs": {
    "type": "name",
    "properties": {
      "name": "urn:ogc:def:crs:EPSG::{EPSG_CODE}"
    }
```

```
  },
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [x_geographic, y_geographic]
      },
      "properties": {
        "dx": displacement_x_pixels,
        "dy": displacement_y_pixels,
        "score": klt_confidence_score,
        "radial error": euclidean_displacement,
        "angle": displacement_angle_degrees,
        "zncc_score": cross_correlation_score
      }
  }
 ]
}
```

The Figure below shows GeoJSON file content over an RGB raster image by using QGIS. The set of KP (depicted in marble colour) as computed by KARIOS are displayed. The set of KP is fetched by clicking on corresponding geometric feature. KP metadata can be retrieved (bottom right figure).

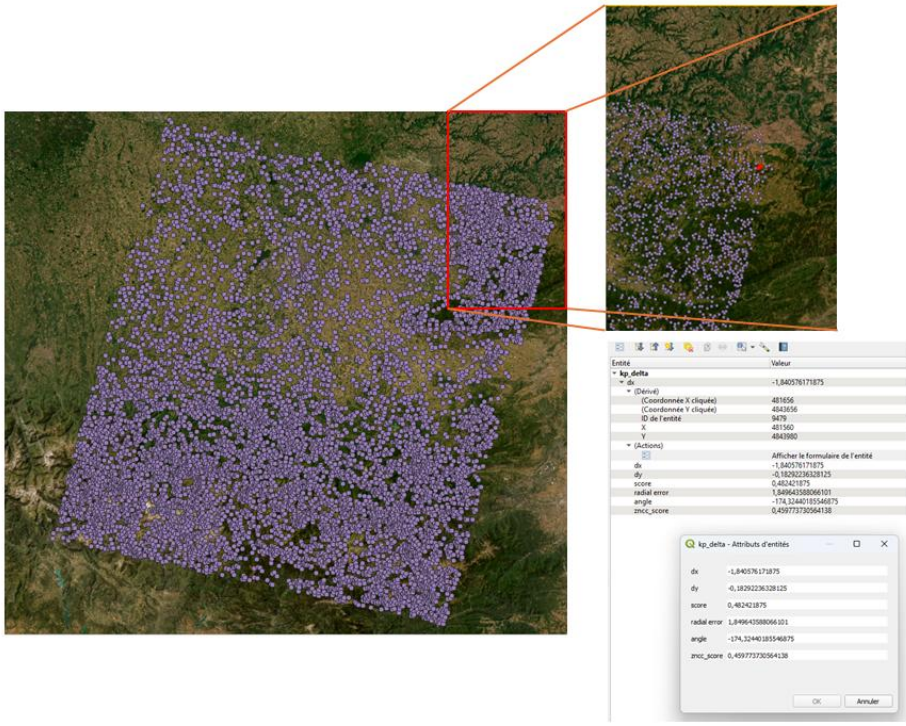


Figure 6-3: KARIOS JSON File opened into QGIS software.

6.2 Visualizations

KARIOS outputs the following figures, ready for geometric accuracy assessment

01_overview.png: Error distribution overview with image thumbnails	§6.2.1 Errors overview
04_ce.png: Circular error analysis with statistical summaries	§6.2.2 Error distribution
02_dx.png: X-direction displacement analysis by row/column	§6.2.3 Disparity maps
03_dy.png: Y-direction displacement analysis by row/column	§6.2.3 Disparity maps
dem_*.png: DEM-based altitude analysis (if DEM provided)	§6.2.4 Dependency on terrain relief

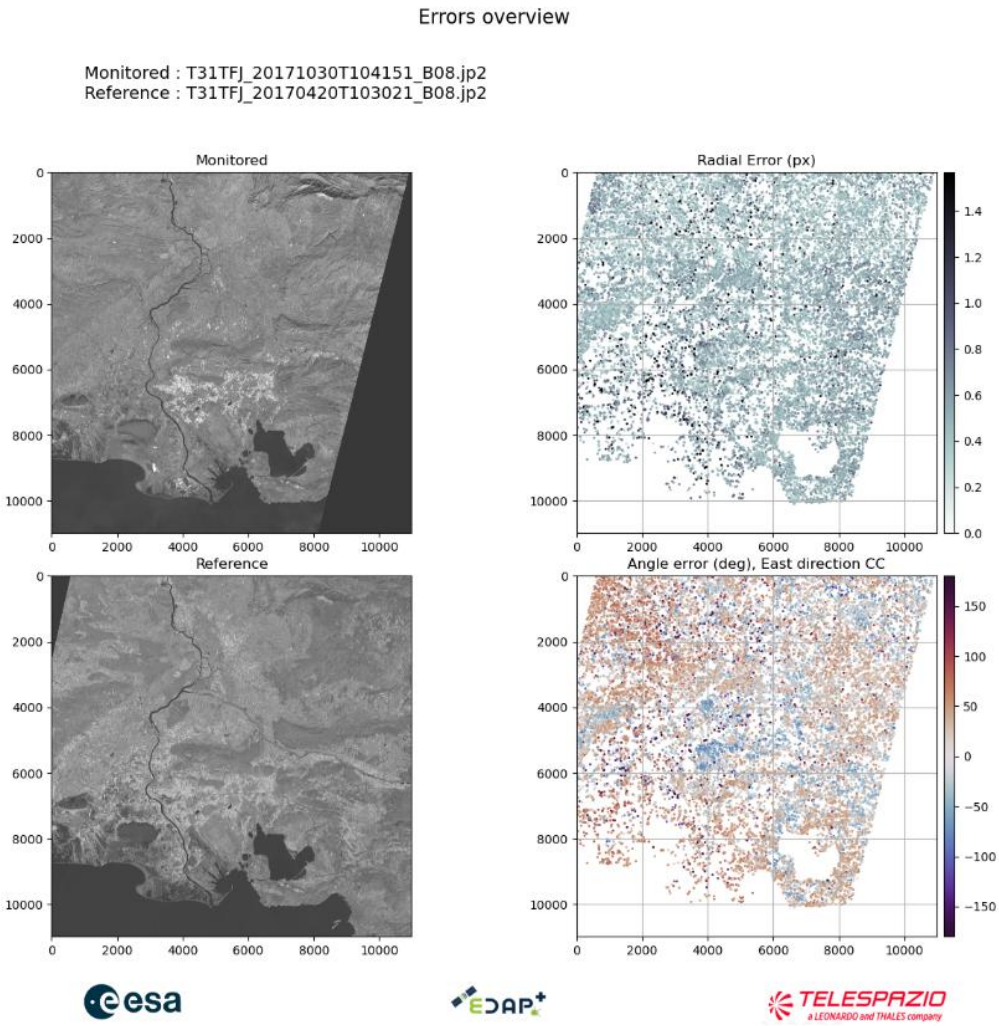
6.2.1 Errors overview

Figure 2-3 displays four image panels detailing input (left) / output (right) images. ‘Monitored’/’Reference’ image names are recalled, in agreement with the convention used in the command line interface. The two output images are:

- The radial error, which is the so-called Euclidean distance between the same point in both input grids (upper image)

- The angle error image, defined as the orientation of displacement vector in the image coordinate system (lower image).

Note that, in Figure 6-4, values in the angle error image (degree) are strongly correlated with terrain relief information.



6.2.2

Figure 6-4: Error overview images: Input images (right) and radial / angle error (left), South of France area (Salon de Provence).

Error distribution

Figure 2-4 displays several graphics enabling the analysis of the one dimensional / two-dimensional distribution of displacement errors in the cartographic coordinate system (Easting (m) / Northing (m). the graphical representation of the cumulative error function (right location) is also given. In addition, accuracy statistics, as indicated with (D), are similar to those defined in Section 2.2.2.

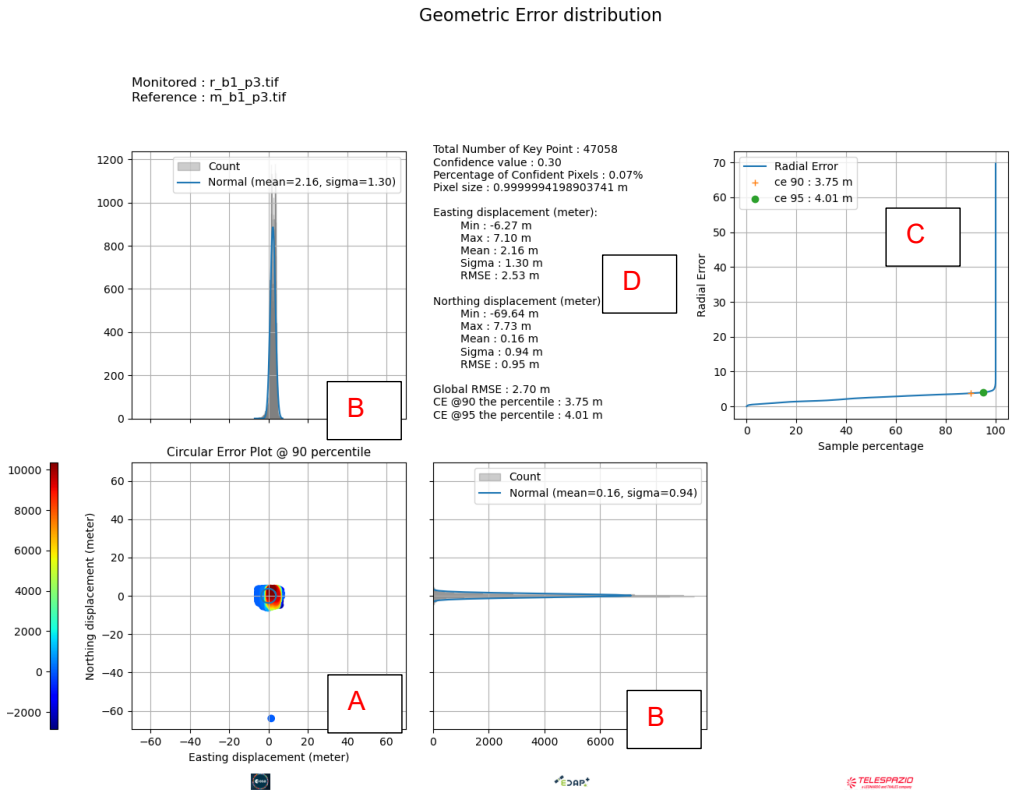


Figure 6-5: Geometric error distribution figure; Circular Error (A), Histogram for each direction (B), 2D Error Cumulative functions (C) and statistics (D).

6.2.3

Disparity maps

This section shows two disparity map figures, corresponding to displacement errors in both pixel (dx) and line (dy) directions. One example of the output figure for displacement in pixel direction (dx) is shown in Figure 6-6 below, showing two graphical representations of the disparity map (image in the centre of the figure).

These two graphic plots reflect mean error values against pixel number (upward graphic plot) and mean error values against line number (rightward graphic plot). For these two plots, the mean error together with the standard deviation (std) errors are depicted in blue. Furthermore, grey bars are also added to indicate the sample number involved ('Number of KP') in the per line / per column statistics.

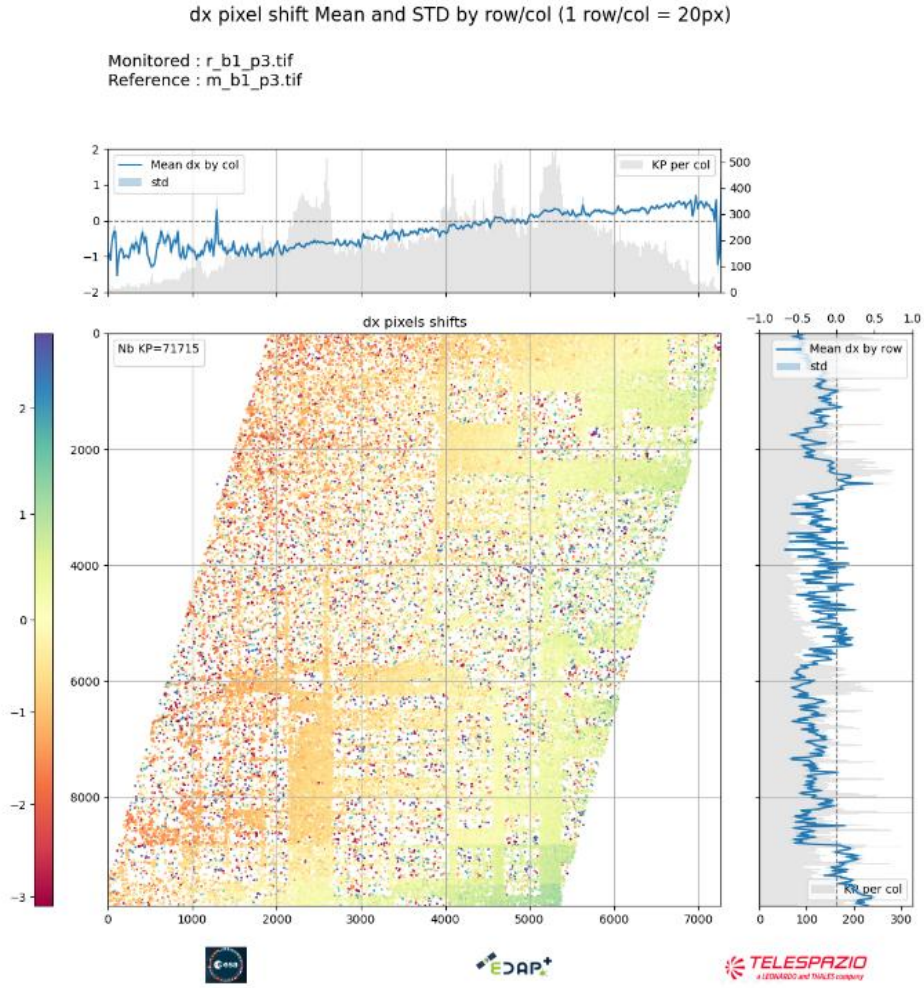
Note that instead of producing per-line / per-column statistics, it is possible to integrate over a group of lines, or a group of columns, as shown where a bin interval of 20 pixels is used.

In Figure 6-6 the observing error across the pixel number (upward graphics) are in range of -1.0 pixel up to 0.5 pixel. In the left / right most part of graphic plot, sample number is very small meaning that statistics are not fully representative. On the other hand, standard deviation all along the pixel number is also very small, indicating a high precision. The colour variation from left to right observed in the image, is confirmed by the upper graphic plot.

Another geometric artefact, which is not observed in the image, becomes visible in the rightward graphic plot where at least two modes of geometric error are observed, one centred on -0.5 pixels and one centred on 0 pixels (refer to line number 6000). This is due

to the geometry of the full image ('monitored'), being built using image composing of small tiles; at the boundaries of each tile, deregistration errors are observed.

As image matching parameters, the format of these graphics are highly configurable (section 9), enabling the user to highlight specific geometric artefacts.



6.2.4

Figure 6-6: Disparity map (in one direction) and mean error in along / across the image.

Dependency on terrain relief

These outputs are proposed to check if the MON image is free from distortions caused by terrain relief. Misregistration errors are analysed depending on altitude interval, altitude values are from the DEM data (optional input),

Accuracy analysis results are presented on the basis of altitude bins. To support this, a box plot representation has been selected. For more information about boxplot please refer to[1]

An example of this output is shown in **Figure 6-7**. The radial error increases with the altitude, and the statistical distribution (within altitude bins) is asymmetric.

[1] https://fr.wikipedia.org/wiki/Bo%C3%AEte_%C3%A0_moustaches

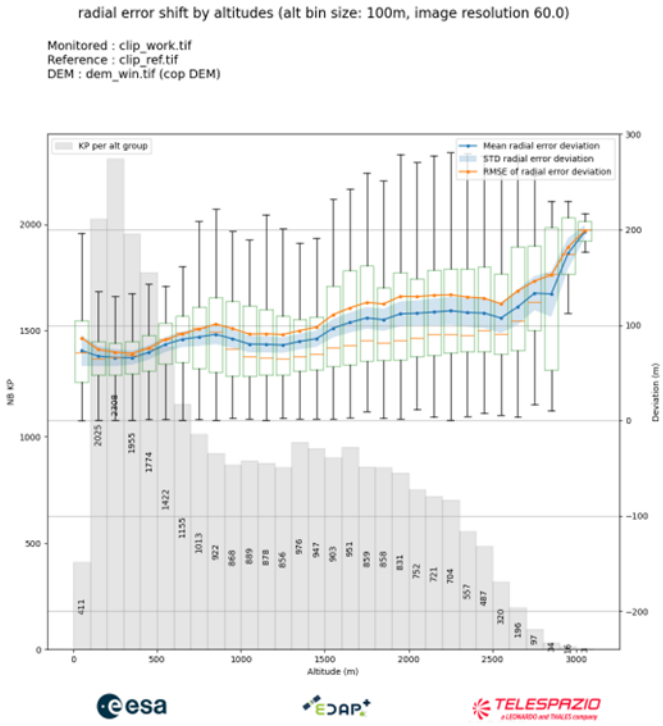


Figure 6-7: Disparity map (in one direction) and mean error in the along / across direction.

6.3 Products

- **kp_mask.tif**: Binary mask of KP locations (if `--generate-key-points-mask` is used)
- **kp_delta.tif**: Two-band raster with dx/dy displacement values (if `--generate-intermediate-product` is used)
- **kp_delta.json**: GeoJSON of key points with displacement vectors (if images are georeferenced). The content of this file is discussed in Section 6.1.2.
- **chips/** directory: Directory containing the KP chip images

6.4 Configuration

A copy of the processing configuration used is part of KARIOS results.

7. PACKAGE INSTALLATION

7.1 Prerequisite

- Python 3.12+
- To run the tool, a dedicated conda environment is required:
  - conda[2]
  - or miniconda[3]
- libGL: you need libGL for Linux, depending on your distribution it can be libgl1-mesa-glx, mesa-libGL or another.

7.2 Get KARIOS

Get the source code,

```
git clone https://github.com/telespazio-tim/karios.git
```

And add it to the karios directory.

```
cd karios
```

7.3 Environment setup

7.3.1 Set up the conda environment

```
conda env create -f environment.yml
```

If the following error message CondaValueError: prefix already exists: <PATH_TO_CONDA_DIR>/envs/karios appears, that means you already have a karios conda environment.

Update it with following command:

7.3.2

```
conda env update -f environment.yml --prune
```

7.3.3 Activate the environment

```
conda activate karios
```

Install KARIOS

For runtime:

```
pip install
```

For development:

```
python -m karios -help
```

or

[2] https://docs.conda.io/projects/conda/en/stable/user-guide/install/index.html

[3] https://docs.conda.io/en/latest/miniconda.html

```
karios --help
```

RUNNING TIME

KARIOS requires two mandatory inputs:

monitored sensor image file

reference sensor image file

8.

Input files shall contain only one layer of data, and the format shall be recognised by the GDAL library.

```
usage: karios.py [-h] [--mask MASK] [--conf CONF] [--out OUT] [--
resume]      [--generate-key-points-mask]      [--input-pixel-size
PIXEL_SIZE] [--no-log-file] [--debug]

                 [--log-file-path LOG_FILE_PATH]

                 mon ref



positional arguments:

  mon                   Path to the monitored sensor product

  ref                   Path to the reference sensor product



options:

  -h, --help            show this help message and exit

  --mask MASK           Path to the mask (default: None)

  --conf CONF                 Configuration file path (default:
/opt/karios/karios/configuration/processing_configuration.json)

  --out OUT                Output results folder path (default:
/opt/karios/results)

  --resume                 Do not run KLT matcher, only accuracy
analysis and report generation (default: False)

  --generate-key-points-mask, -kpm

                        Generate a tiff mask based on KP from KTL
(default: False)

  --generate-intermediate-product, -gip

                        Generate a two-band tiff based on KP with
band 1 dx and band 2 dy (default: False)

  --input-pixel-size PIXEL_SIZE, -pxs PIXEL_SIZE
```

```
                        Input image pixel size in meter. Ignored if
image resolution can be read from input image (default: None)

  --no-log-file         Do not log in file (default: False)

  --debug, -d           Enable Debug mode (default: False)

  --log-file-path LOG_FILE_PATH

                        Log       file       path       (default:
/opt/karios/karios.log)
```

CONFIGURATION

The default configuration parameters are located in karios/configuration/processing_configuration.json

klt_matching parameters

9.

The following parameter allows the user to control how to find the most prominent corners in the reference image, as described by the OpenCV documentation, after the Laplacian process.

9.1

- minDistance : Minimum possible Euclidean distance between the returned corners
- blocksize : Size of an average block for computing a derivative covariation matrix over each pixel neighbourhood, default 3
- maxCorners : Maximum number of corners to extract. If there are more corners than are found, the strongest of them is returned. maxCorners <= 0 implies that no limit on the maximum is set, and all detected corners are returned.
- qualityLevel : Parameter characterising the minimal accepted quality of image corners. The parameter value is multiplied by the best corner quality measure. The corners with the quality measure less than the product are rejected. For example, if the best corner has the quality measure = 1500, and the qualityLevel=0.01, then all the corners with the quality measure less than 15 are rejected.

Other parameters

- matching_winsize : size of the search window during matching corners in the reference and the monitored Laplacian images.
- xStart : image X margin to apply (margin is skipped by the matcher)
- tile_size : tile size to process by KTL in the images.
- laplacian_kernel_size : Aperture size used to compute the second-derivative filters of the Laplacian process
- outliers_filtering : whether to filter or not the outlier points found during the matching.

It is worth noting that this program is calling CV2 function calcOpticalFlowPyrLK. In the KARIOS implementation, the default value for the parameters is kept. It deals with the following parameters:

9.2

- Size winSize = Size(21, 21), size of the search window at each pyramid level
- int maxLevel = 3, number of levels in the pyramid

Please refer to Section 9.6 for discussion on parameter tuning.

accuracy_analysis

- confidence_threshold : max score for points found by the matcher to use to compute statistics written in correl_res.txt. If “None”, not applied

plot_configuration.overview

- fig_size : Size of the generated figure in inches
- shift_colormap : matplotlib colour map name for the KP shift error scatter plot
- shift_auto_axes_limit : auto compute KP shift error colourbar scale
- shift_axes_limit : KP shift error colourbar maximum limit, N/A if shift_auto_axes_limit is true
- theta_colormap : matplotlib colour map name for the KP theta error scatter plot

9.3

plot_configuration.shift

- fig_size : Size of the generated figure in inches
- scatter_colormap : matplotlib colour map name for the KP shift scatter plot
- scatter_auto_limit : auto compute KP shift scatter plot limit
- scatter_min_limit : KP shift scatter plot minimum limit, N/A if scatter_auto_limit is true
- scatter_max_limit : KP shift scatter plot maximum limit, N/A if scatter_auto_limit is true
- histo_mean_bin_size : KP shift histogram bin size (number of image row/col for the histogram bin)

9.4

plot_configuration.ce

- fig_size : Height size of the generated figure in inches, width is 5/3 of the height
- ce_scatter_colormap : matplotlib color map name for the KP shift density scatter plot

9.5

KLT parameter leverage

9.6

For many reasons, it is important to carefully setup the configuration file. For doing this, the following key aspects should be considered.

1. Context: Calibration, Validation, Performance monitoring
2. Assessment Items: Absolute, inter band, multi temporal
3. Image instrument types: Optical, Radar, Thermal, and whether comparing images from the same instrument. The same instrument type is more straightforward than comparing images from different instrument types (e.g., Optical vs Radar)
4. Spectral band, when relevant
5. Image size

9.6.1

maxCorners & tile_size

To minimise memory usage during KLT processing, it is possible to define a tile size to process for KLT.

For example, to process in an image with a size of 20000x20000 pixels, processing time will be saved by adopting a tile_size of 10000 . It will result in four tiles to be processed. and the matching process is run four times,

In this context, the KLT process will look in each tile for maxCorners.

While an image of 20000x20000 pixels results in 4 equal tiles, an image of 20000x15000 pixels also results in 4 tiles, but with a different size: two of 10000x10000 pixels and two of 10000x5000 pixels.

The consequence is that the density for matching points will not be the same each tile, the bigger tiles will have a lower number of matching points than the smallest.

You may also consider that the image can contain empty parts where KLT will not find any matching points. So, tiles having a large empty parts will also result in areas with higher matching point density.

In order to avoid density differences in the final result, you can define a `tile_size` larger than the image with a high `maxCorners`, or a small `tile_size` and `maxCorners` .

For example, for image of 20000x15000 pixels, you should consider a `tile_size` of 20000 (1 tile), or 5000 (12 equal tiles).

9.6.2 Min distance & Matching Window size

To collect many KPs with a risk of increasing the number of outliers, the following approach should be adopted:

- Remove outlier filtering
- Reduce min distance
- Increase max corner.
- Increase matching window size

Getting many KPs is useful to characterise deformation, for instance, in this case, disparity map curves (dx dy profiles) become smoother as they include less noise, which is required for calibration. Nonetheless, this configuration is useless if the primary objective is performance monitoring of the geolocation accuracy.

10. RESOURCES

- KARIOS Documentation (Poster), https://zenodo.org/records/10598329
- KARIOS S/W, https://github.com/telespazio-tim/karios, (git clone: https://github.com/telespazio-tim/karios.git
- QUICK START https://telespazio-tim.github.io/karios/quickstart.html
- Landing Page, https://telespazio-tim.github.io/karios/

[END OF DOCUMENT]