



SIFMap: A python toolbox for creating solar-induced fluorescence maps from UAV-borne snapshot imaging data

J. Buffat^{a,*}, A. Elibol^b, H. Scharr^a, S. Choza-Farías^c, S. Salattna^c, J. Bendig^c

^a Forschungszentrum Jülich GmbH, Institute of Advanced Simulations, IAS-8: Data Analytics and Machine Learning, Jülich, Germany

^b College of Engineering, Al Ain University, Abu Dhabi Campus, United Arab Emirates

^c Forschungszentrum Jülich GmbH, Institute of Bio- and Geosciences, IBG-2: Plant Sciences, Jülich, Germany

ARTICLE INFO

Keywords:

Remote sensing
Photosynthesis
Mosaic
Solar-induced fluorescence
Uncrewed aerial vehicles
SIFcam

ABSTRACT

Solar-induced chlorophyll fluorescence (SIF) is a rich source of information for understanding plant photosynthetic processes, carbon assimilation, functional status, and early stress symptoms of plants. As plants assimilate sunlight, they do so across the spectral range of photosynthetically active radiation. Subsequently, during photosynthesis, the excess solar energy is radiated as heat and emitted as chlorophyll fluorescence in the red to far-red spectral range. Recently, SIFcam, a novel drone-based snapshot and filter-based camera array tuned to retrieve SIF at the centimeter scale has been introduced. The absence of transparency regarding methodologies and algorithms in commercial map creation software complicates the investigation of the impacts of various parameter configurations in the SIFcam processing chain. Consequently, we have engineered a Python toolbox that replicates the primary features of commercial software in a transparent and adaptable fashion. To this end, we follow a feature-based image mosaicking pipeline and present modifications to create SIF maps. Moreover, it provides detailed and insightful information while being fully controllable by the user. The toolbox comprises different primary functions, with each function corresponding to a specific stage within the map generation pipeline. The modular design of the toolbox facilitates straightforward adjustments or substitutions of individual steps.

Metadata

Code metadata.

Nr.	Code metadata description	Please fill in this column
C1	Current code version	v2.0.0
C2	Permanent link to code/repository used for this code version	https://github.com/IAS-8/SIFMap
C3	Permanent link to Reproducible Capsule	None
C4	Legal Code License	GPL 3.0 or later
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	python, OpenCV
C7	Compilation requirements, operating environments & dependencies	see requirements.txt in software package
C8	If available Link to developer documentation/manual	see README.md and CONFIGURATION.md in repository
C9	Support email for questions	ias8-support@fz-juelich.de

1. Motivation and significance

Solar-induced fluorescence (SIF) is a light flux emitted in the far-red spectral range by chlorophyll absorbing sunlight across the

photosynthetically active spectrum. The fraction of absorbed energy that is re-emitted as fluorescence is informative of a plant's photosynthetic activity [1,2]. In the last decade, many sensors on airborne

* Corresponding author.

Email address: j.buffat@fz-juelich.de (J. Buffat).

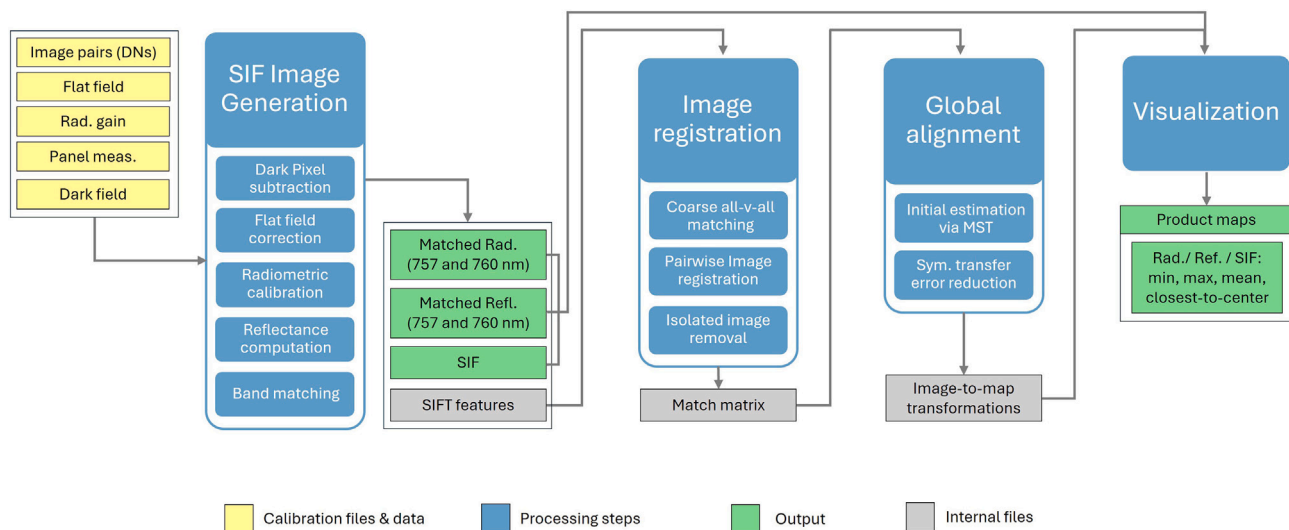


Fig. 1. Overall pipeline of SIFMap modules along with the main processing functionalities. DNs: digital numbers, Rad.: radiance, Refl.: reflectance, meas.: measurements, all-v-all: all versus all, Sym.: symmetric, MST: minimum spanning tree.

and spaceborne platforms have been tested for SIF retrieval at local to global scales [3,4]. Remote sensing of SIF from Uncrewed Aerial Vehicles (UAVs) has so far been dominated by works on point spectrometers [5] and mid-resolution imaging sensors for relative SIF retrieval [6].

In this contribution, we present SIFMap, a custom-tailored feature-based preprocessing and mosaicking pipeline to generate SIF maps at 760 nm from data collected by a new high-resolution imaging sensor called SIFcam [7]. SIFcam carries two cameras fitted with ultra-narrowband interference filters (central wavelengths: 757.9 nm and 760.7 nm) to allow for a Fraunhofer Line Discrimination (FLD) retrieval [8]. Operating at a typical altitude of 25–30 meters, the SIFcam system provides data to generate SIF images with high spatial resolution (in the order of centimeters).

SIFcam is comparable to a multispectral camera array. Acquired data can be processed in software other than SIFMap, such as Agisoft Metashape [9], Pix4Dmapper, and WebODM [10] (see runtime comparison in Table 1). However, the absence of transparency around core algorithms in commercial software makes it challenging to investigate the impacts of various parameter configurations on the radiometry in the output product. Agisoft Metashape and Pix4Dmapper offer some control over pixel blending options, interpolation methods and minor corrections but do not allow insight into the final, proprietary seamline generation algorithm. In comparison, WebODM provides some control over blending options, allows customization of seamlines, and enables control of cutline behaviors, but is not adapted to the large number of images of a typical SIFcam flight resulting in unsustainable processing times as it relies on the Structure-from-Motion (SfM) pipeline [11], which is generally more complicated and computationally intensive than dedicated 2D mosaicking approaches [12,13]. Furthermore, WebODM does not recognize SIFcam as a multispectral camera array, allowing only one spectral band to be processed at a time. As a result, we developed a fully customizable Python toolbox that replicates the core functionalities of commercial software but with full transparency and adaptability providing pixel-level statistics.

The toolbox comprises distinct functions, each aligned with a specific stage in the map generation pipeline. This design facilitates adjustments or substitutions of individual steps. The toolbox's key features include:

- Full implementation in Python using numpy, SciPy, scikit-image, and OpenCV
- Object-oriented architecture with parallel processing enabled by numba and joblib

- A modular design for easy modification based on specific needs (e.g., generating maps from RGB images)
- Automatic removal of isolated or corrupted images/data
- Initial image-to-map estimates derived from graph theory methodologies
- Comprehensive error minimization, including analytical Jacobian computations, sparsity pattern analysis and interactive outlier analysis
- Direct access to pixel-level SIF values and statistics

2. Software description

We follow a feature-based mosaicking pipeline as outlined in previous studies [14–16] and present modifications to create SIF maps computed according to the FLD method [7]. The software is written in Python utilizing the numpy, SciPy, scikit-image, and OpenCV libraries. The main modules, data flow, and connections between modules are detailed in Fig. 1.

2.1. Software architecture

SIFMap implements an adaptable, open-source processing solution for SIFcam imagery by performing (i) FLD SIF retrieval as in [7], (ii) image registration, (iii) global alignment and (iv) the provision of reflectance and SIF mosaics that can be geo-registered with third-party software for further downstream analysis. To this end, the SIFMap toolbox is composed of three main modules (SIFcam, match, align). A fourth module (data) defines the basic data structures that interface with the preprocessing and mapping logic of SIFMap. SIFMap is formulated in a generalized fashion such that the alignment and registration functionalities can easily be applied to any image data type organized into classes derived from data.

Preprocessing In this step, raw images of each channel are converted into radiometrically calibrated at-sensor radiance and subsequently atmospherically corrected to reflectance products at 757 and 760 nm (Ref₇₅₇ and Ref₇₆₀). Flat field correction files, pre-flight dark-field acquisitions, radiometric gains as well as in-flight measurements of Lambertian reflectance panels are needed as input for this module. The radiance is calculated by applying a dark field subtraction, flat field correction, and scalar radiometric calibration (assuming a perfectly linear response). The reflectance is obtained

Table 1

Execution time (s) and peak RAM usage (GB) of the processing pipelines Agisoft [9], SIFMap and WebODM [10] applied to the datasets published along with this contribution [19] on a 32-core machine. For WebODM we only report a global run time due to a lack of software reporting. N: number of SIFcam image pairs, Preproc.: Preprocessing, Registr.: Registration, Align.: Alignment, Vis.: Visualization, Total map.: Total time for mapping (excluding preprocessing). The *visualization* step in the case of SIFMap refers to the run time for reflectance and SIF products renormalized to the time for a single product (divided by three), in the case of Agisoft and WebODM it reports the run time for a single raw image orthomosaic. The 21/06/13 dataset consists of 1024×1024 pixel images, and the 24/06/08 dataset of 512×512 pixel images.

Dataset	Method	N	Preproc.		Registr.		Align.		Vis.		Total map.	
			[s]	[GB]	[s]	[GB]	[s]	[GB]	[s]	[GB]	[s]	[GB]
21/06/13	SIFMap	342	167	18	77	0.35	170	0.65	240	146	487	146
	Agisoft		–	–	190	1.25	130	1.52	140	2.41	460	2.41
	WebODM		–	–	–	–	–	–	–	–	7'213	–
24/06/08	SIFMap	931	131	13	418	0.91	336	0.26	158	100	912	100
	Agisoft		–	–	311	0.63	497	1.37	96	0.59	904	1.37
	WebODM		–	–	–	–	–	–	–	–	7'728	–

by linear interpolation of the radiance over the panel measurements. Equally, we compute the irradiance by extrapolating the reflectance panel measurements to a perfectly reflective surface.

The functionalities of this preprocessing step are organized in the file located at `SIFcam/data.py`.

Band alignment A scene can be approximated as planar when the distance between the camera and the scene is significantly greater than the depth variations of the scene. In this case, dense correspondence mapping can be achieved through estimation of a homography between the two channels (see [Appendix A](#)). The planarity assumption is sufficiently fulfilled in SIFcam flight heights larger than 25 m over typical crops forming closed canopies (e.g., wheat). As highlighted below, SIF is computed by combining individual pixels in both image channels, requiring subpixel matching accuracy. As a consequence, the homography is estimated for each image pair even though the SIFcam camera system is fixed. This allows adjusting for vibration and speed variations. The latter affects the matching differently across image pairs due to different integration times in the two channels. The homography estimation is implemented as a Random Sample Consensus (RANSAC) [17] based matching of Scale Invariant Feature Transform (SIFT) [18] keypoints. Finally, to compute SIF from the aligned reflectance channels and the estimated irradiance at 757 and 760 nm, we use FLD in its most basic formulation [8]

$$f_{760} = \frac{(\text{Ref}_{760} - \text{Ref}_{757}) \cdot \text{Irr}_{757} \cdot \text{Irr}_{760}}{\text{Irr}_{757} - \text{Irr}_{760}} \quad (1)$$

Image Registration Module In this step, spatial relationships between images are detected. First, a list of possibly overlapping image pairs is generated by comparing a randomly selected set of feature descriptors in an all-against-all manner with a nearest-neighbour approach. These coarsely identified matches are verified by accurately matching the image pairs using projective descriptor matching and RANSAC. Filtering the coarse matches provides us with a graph of connected images. Finally, we remove isolated images from the topology.

The functionalities of this module are organized in the files located in the `match` module. `register.py` contains the algorithms necessary to obtain and register possibly overlapping image pairs.

Global Alignment In this step, transformations for each image are established in order to map them onto a single common coordinate frame. This is done by minimizing the non-linear symmetric transfer error [20] with a least-squares optimization. To gain an initial estimate we find the image using the minimum spanning tree in the matching graph. This image is selected as the global frame. The

shortest paths from this image to all other images are computed and pairwise motions along the path are accumulated. Once this step is finalized, map-to-image (and image-to-map) transformations are derived as minima of the transfer error.

In order to address noisy SIFT feature matches from the registration step, the alignment is run in several least squares optimizations with iterative outlier removal. Before each new optimization run, the quality of feature pairs under the current motion model is compared and outlier correspondences are rejected.

The functionalities of this module are organized within the `align` module. `align.py` includes functions that compute image-to-map transformations taking into account the user-selected parameters. `jacobians.py` contains functions to compute Jacobian matrices derived using analytic computations and functions to compute their sparsity patterns.

Visualization This step visualizes stitched maps of SIF and reflectance. Since image-to-map transformations are now known for every pixel of the map, different products can be generated depending on variable aggregation strategies of overlapping images. The aggregation reduces the distributions of overlapping images (see [Fig. 2](#)) to single estimates. Our module currently provides minimum, maximum and mean values of these distributions. Furthermore, we provide products taking into account the distance from a pixel to the center of individual images: with the *closest* aggregation method the image with the shortest distance to a pixel is selected. With the *distance-weighted* aggregation method different image contributions are weighted according to d^{-2} , with d being the pixel-center distance. This produces smoothed products reducing the impact of changing lighting as a consequence of variable sun-observer-geometry. At the moment, no further analysis is applied in this step, such as a radiometric normalization or artefact characterization.

The functionalities of this module are organized in the file `SIFcam/visual.py`. It includes the function `draw` which encompasses the parallelized aggregation of all images in a global map.

2.2. Software functionalities

2.2.1. Code performance

SIFcam datasets are typically acquired at flight heights of 25–30 meters (with 80 % forward overlap and 70 % sidelap consisting of 300–1000 image pairs) [7,16]. For a detailed validation study of SIFcam products we refer the reader to [16]. In the following, we report stage-wise SIFMap code execution time scales (see [Table 1](#)) for two datasets made available alongside this code publication [19].

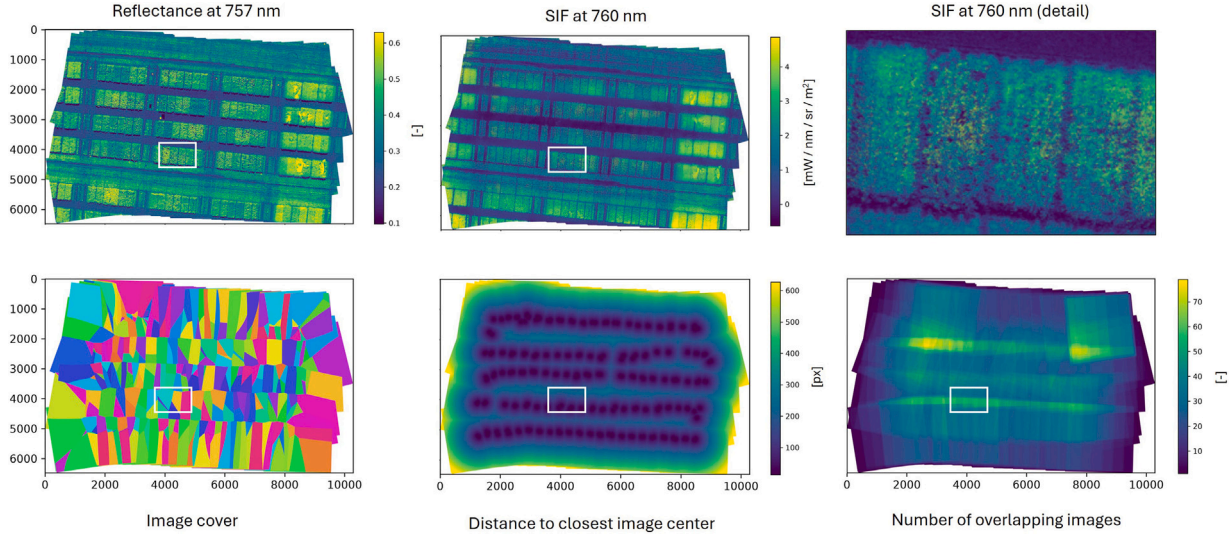


Fig. 2. Sample quick looks generated by SIFMap of the 21/06/13 data set [19]. Upper row: Shown are the reflectance at 757 nm and SIF at 760 nm aggregated with `closest_distance_weighted` and a detailed view of SIF. The white rectangle highlights its position. Lower row: Shown are the image coverage by the closest image, the distance to the closest image and the number of covering images per pixel.

2.2.2. User configuration

We provide detailed instructions on the correct usage of SIFMap in the code package. To guide the user, we provide (i) a small example data set in the code package (see Section 3) and (ii) two data sets consisting of complete SIFcam flights [19]. SIFMap's preprocessing step features a limited set of hyperparameters with limited impact on the matching and alignment of SIFcam images acquired at > 25 m. The registration and alignment steps exhibit a larger number of hyperparameters. While the default configuration has been observed to yield consistently acceptable results in comparable SIFcam datasets, data set dependent changes might be necessary on occasion. In the following, we highlight the most important user configuration choices regarding these steps making use of the naming convention defined in the code package. `CONFIGURATION.md` in the code package provides a more complete parameter description.

`registration.ransac_residual_threshold`: This value determines the quality of feature matches returned by the *registration* step. It is given as the allowed pixel mismatch for a correspondence under a RANSAC optimized projective transform. A lower threshold leads to a smaller number of accepted correspondences. A larger value is suitable for datasets with changing lighting, but introduces more low quality correspondences.

`registration.ransac_residual_threshold`: This value determines the quality of feature matches returned by the *registration* step. It is given as the allowed pixel mismatch for a correspondence under a RANSAC optimized projective transform. A lower threshold leads to a smaller number of accepted correspondences. A larger value is suitable for datasets with changing lighting, but introduces more low quality correspondences.

`registration.min_n_correspondences`: This value determines the minimum number of correspondences between two images in order to consider them connected. Remaining correspondences are eliminated once their number drops below this value.

`alignment.manual_outlier_mode`: Setting this parameter to true allows dynamically continuing/stopping the optimization.

`alignment.n_outlier_removal`: This parameter determines the number of least squares optimizations and outlier removals to be run iteratively.

`alignment.n_points`: This value determines the maximum number of correspondences in image pairs considered during the loss computation. If this value is chosen much larger than `registration.min_n_correspondences` the chance of dropping a connection between images decreases.

`alignment.outlier_removal`: This value determines which quality of correspondence is considered to be insufficient (i.e., an *outlier*) after each least squares optimization. It is given in terms of the standard deviation of the feature residual distribution. Higher values lead to less *outlier* detection, while smaller values lead to more aggressive correspondence pruning.

`alignment.optim_params`: The parameters provided in this dictionary structure guide the least-squares optimization.

3. Illustrative example

We provide a small testing dataset consisting of 24 image pairs in `example_data.01_mixed_crops_13062021` and two data sets featuring complete SIFcam flights with 342 and 931 image pairs [19]. The data sets include the configuration files `config.yaml` and `sensor_calib.yaml` as example parameterizations of the preprocessing, the matching, the alignment optimization and the visualization.

The full SIFMap pipeline is executed by starting the main script as detailed in the software package. To run the testing data set, follow the installation steps to set up your Python and user environment and call

```
bash docker/run_wo_docker.sh
-c ../user_env.sh
example_data/01_mixed_crops_13062021/calib.yaml
```

Listing 1. The most basic command to call SIFMap on the test case. See `README.md` in the software package for more detailed explanations of proper SIFMap usage.

The program first prints the configuration parameters and continuously logs the optimization. If the user runs with outlier removal (i.e., with `alignment.n_outlier_removal > 0`) the optimization statistics are printed for multiple iterations. Finally, quick looks of SIF, reflectance and mapping information (examples are provided in Fig. 2) as well as the full data are saved in the output folder.

4. Impact

Remote sensing of SIF is a rapidly evolving technique for quantifying photosynthetic activity and associated stress in vegetation. This approach enables the assessment of physiological processes at various scales, from individual leaves to entire ecosystems, leveraging both in-situ measurements and satellite-derived products. Given the relevance of chlorophyll fluorescence, it is of great interest to develop technologies to obtain high-resolution SIF maps in a low-cost fashion and with minimal manual intervention. The SIFMap processing software provides a versatile code base for preprocessing, SIF estimation and mosaicking of SIFcam [7] acquisitions. Currently, the SIFMap software is restricted to SIFcam data sets from flights at heights > 25 m over crop-like vegetation canopies. This is due to the homography-based matching of the two channels which assumes planar features. Possible solutions to address matching in data sets with closer, more complex targets are being outlined by the development team and will be published in future versions.

5. Conclusions

Existing airborne and ground-based sensors for measuring and obtaining SIF data come with different limitations (e.g., resolution, operational costs, deployment, and similar factors). UAVs equipped with relatively low-cost sensor setups enable flexible, high spatial resolution data acquisition. To enrich their usability, adapted software solutions (such as mapping) are beneficial and needed. With this software package we provide a flexible end-to-end mapping toolbox in Python to create 2D SIF maps from SIFcam imaging data tested for flat terrain applications. Extensions for additional SIF retrieval and mosaicking methods are possible in future feature implementations. Thanks to its modular design, SIFMap is customizable and profits from parallel programming paradigms to reduce the processing time.

CRedit authorship contribution statement

J. Buffat: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **A. Elibol:** Writing – original draft, Visualization, Validation, Supervision, Software, Methodology, Investigation, Data curation, Conceptualization. **H. Scharr:** Supervision, Resources, Project administration, Funding acquisition. **S. Choza-Farías:** Writing – review & editing, Visualization, Validation, Data curation, Conceptualization. **S. Salattna:** Visualization, Validation, Data curation. **J. Bendig:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Project administration, Methodology, Funding acquisition, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We thank the team at Campus Klein-Altendorf (University of Bonn) for growing the experimental crops. We gratefully acknowledge the ‘Strukturwandel-Projekt BioökonomieREVIER,’ funded by the German Federal Ministry of Education and Research, for supporting the sensor development. Open access to this article is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 491111487. S. Salattna acknowledges funding for this project from the Palestinian-German Science Bridge financed by the German Federal Ministry of Education and Research (BMBF). The authors thank Christian Hirt for his technical support.

Appendix A. Homography-based image matching and alignment

Planarity Assumption A scene can be approximated as planar when the camera undergoes translation, rotation, and/or scaling, provided that the distance between the camera and the scene is significantly greater than the depth variations of the scene. Under these conditions, surfaces that are not oriented toward the camera’s optical axis become scarcely visible and, in extreme cases, may degenerate into thin line-like structures across different views. Consequently, interest points detected on such surfaces are typically identified as inconsistent correspondences and rejected during the outlier filtering stage.

A.1. Planar homography

Motion between two different views of the same flat scene can be described by means of a *planar homography matrix* [20,21]. Let us consider a point p , belonging to a 2D plane Π in 3D space. Then, the projections of p in two different images I_1 and I_2 are given by $(\mathbf{x}_1, \mathbf{x}_2)$. Let the coordinate transformation between the two frames also be

$$\mathbf{X}_2 = R\mathbf{X}_1 + T \quad (\text{A.1})$$

where $\mathbf{X}_1, \mathbf{X}_2$ are the coordinates of p relative to the camera frames of the images I_1 and I_2 respectively. The two projections $\mathbf{x}_1, \mathbf{x}_2$ of p satisfy the epipolar constraint

$$\mathbf{x}_2^T E \mathbf{x}_1 = \mathbf{x}_2^T \hat{T} R \mathbf{x}_1 = 0 \quad (\text{A.2})$$

However, for points on the same plane Π , their images will share an extra constraint that makes the epipolar constraint alone no longer sufficient. Let $N = [\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3]^T \in \mathbb{S}^2$ be the unit normal vector of the plane Π with respect to the first camera frame, and let $d > 0$ denote the distance from the plane Π to the optical center of the first camera. Then we have

$$N^T \mathbf{X}_1 = \mathbf{n}_1 X + \mathbf{n}_2 Y + \mathbf{n}_3 Z = d \Leftrightarrow \frac{1}{d} N^T \mathbf{X}_1 = 1, \quad \forall \mathbf{X}_1 \in P \quad (\text{A.3})$$

Substituting Eq. (A.3) into Eq. (A.2) gives

$$\mathbf{X}_2 = R\mathbf{X}_1 + T = R\mathbf{X}_1 + T \frac{1}{d} N^T \mathbf{X}_1 = \left(R + \frac{1}{d} T N^T \right) \mathbf{X}_1 \quad (\text{A.4})$$

We define the matrix H as follows

$$H = R + \frac{1}{d} T N^T \in \mathbb{R}^{3 \times 3} \quad (\text{A.5})$$

It is referred to as the (*planar*) *homography matrix*, since it denotes a linear transformation from $\mathbf{X}_1 \in \mathbb{R}^3$ to $\mathbf{X}_2 \in \mathbb{R}^3$ as

$$\mathbf{X}_2 = H\mathbf{X}_1 \quad (\text{A.6})$$

Note that the matrix H depends on the motion parameters R, T as well as the structure parameters N, d of the plane P . Due to the inherent scale ambiguity in the term $\frac{1}{d} T$ in Eq. (A.5), one can at most recover from H the ratio of the translation T scaled by the distance d . From

$$\lambda_1 \mathbf{x}_1 = \mathbf{X}_1, \quad \lambda_2 \mathbf{x}_2 = \mathbf{X}_2, \quad \lambda_2 \mathbf{x}_2 = H\mathbf{X}_1 \quad (\text{A.7})$$

We have

$$\lambda_2 \mathbf{x}_2 = H \lambda_1 \mathbf{x}_1 \Leftrightarrow \mathbf{x}_2 \sim H \mathbf{x}_1 \quad (\text{A.8})$$

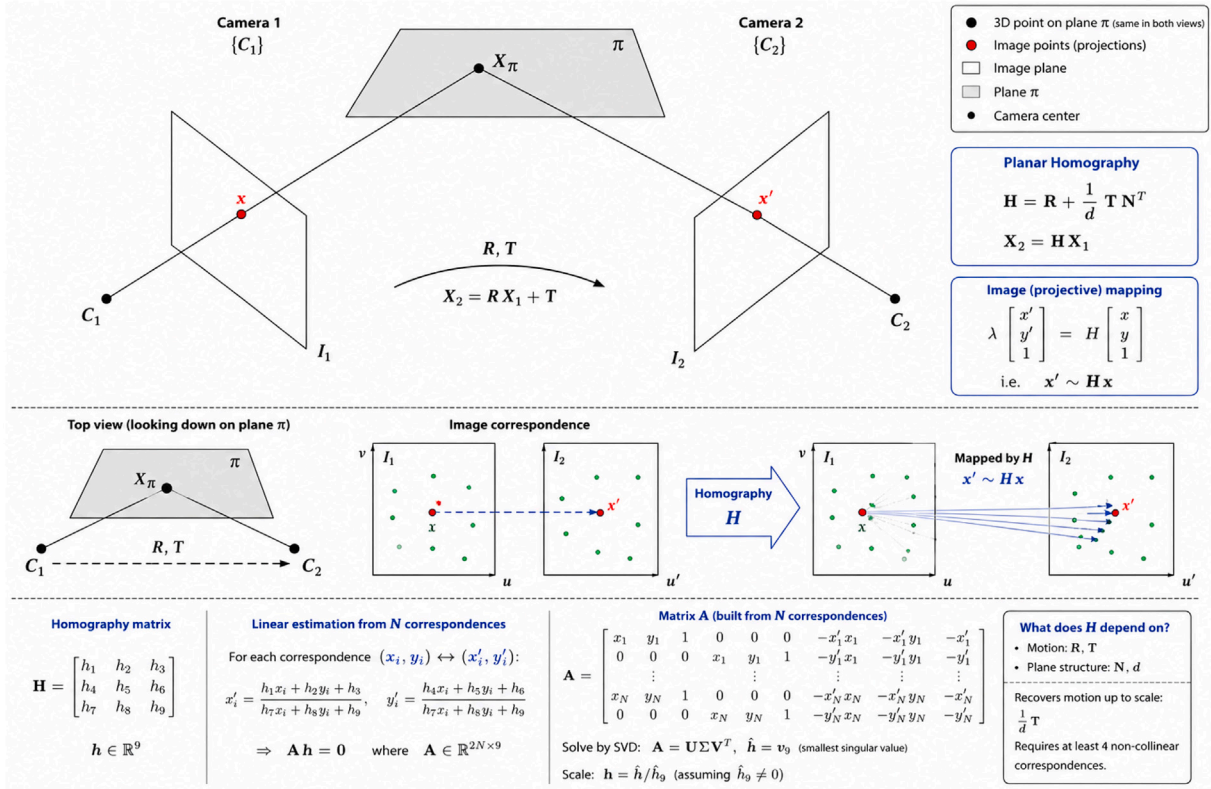


Fig. A.3. Outline of homography matching adopted for SIFmap channel alignment assuming planar geometry.

where we recall that $\sim$ indicates equality up to a scale factor. Often, it is represented as follows:

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

$$\lambda \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

The homography matrix H encodes information about the camera motion and the scene structure, a fact that facilitates establishing correspondence between points in the first and second images. H can in general be computed from a small number of corresponding image pairs.

If h denotes the vector form of H , the above projective homography can be expressed as two linear constraint equations on h . If we have N correspondences $\{X_{1i}, X_{2i}\} (i = 1 : N)$, where $X_{1i} = (x_i, y_i)$ and $X_{2i} = (x'_i, y'_i)$, N constraint equations can be written as $Ah = 0$, where

$$A = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N & y_N & 1 & 0 & 0 & 0 & -x'_N x_N & -x'_N y_N & -x'_N \\ 0 & 0 & 0 & x_N & y_N & 1 & -y'_N x_N & -y'_N y_N & -y'_N \end{bmatrix} \quad (A.9)$$

Let $A = U \Sigma V^T$ denote the singular value decomposition of A . Equivalently, we consider the eigenvalue decomposition $Q = A^T A = V D^2 V^T$, where $\Sigma = D^2$. The linear solution of the 9×1 unit homography vector $\hat{h}$ is the eigenvector associated with the smallest singular value: $\hat{h} = v_9$ (diagonal elements of Σ are arranged in descending order, and v_i is the i -th column of V). Subsequently, h can be determined by scaling: $h = \hat{h} / \hat{h}_9$ (assuming that $\hat{h}_9 \neq 0$) (Fig. A.3).

References

- [1] Frankenberg C, Berry J. Solar induced chlorophyll fluorescence: origins, relation to photosynthesis and retrieval. In: Comprehensive remote sensing. Elsevier; 2018. p. 143–62. <https://doi.org/10.1016/B978-0-12-409548-9.10632-3>
- [2] Sun Y, Gu L, Wen J, van der Tol C, Porcar-Castell A, Joiner J, Chang CY, Magney T, Wang L, Hu L, Rascher U, Zarco-Tejada P, Barrett CB, Lai J, Han J, Luo Z. From remotely sensed solar-induced chlorophyll fluorescence to ecosystem structure, function, and service: Part I—Harnessing theory. Glob Change Biol 2023;29(11):2926–52. <https://doi.org/10.1111/gcb.16634>
- [3] Mohammed GH, Colombo R, Middleton EM, Rascher U, van der Tol C, Nedbal L, Goulas Y, Pérez-Priego O, Damm A, Meroni M, Joiner J, Cogliati S, Verhoef W, Malenovsky Z, Gastellu-Etcheberry J-P, Miller JR, Guanter L, Moreno J, Moya I, Berry JA, Frankenberg C, Zarco-Tejada PJ. Remote sensing of solar-induced chlorophyll fluorescence (SIF) in vegetation: 50 years of progress. Remote Sens Environ 2019;231:111177. <https://doi.org/10.1016/j.rse.2019.04.030>
- [4] Bandopadhyay S, Rastogi A, Juszczak R. Review of top-of-canopy sun-induced fluorescence (SIF) studies from ground, UAV, airborne to spaceborne observations. Sensors 2020;20(4):1144. <https://doi.org/10.3390/rs20041144>
- [5] Vargas JQ, Bendig J, Mac Arthur A, Burkart A, Julitta T, Maseyk K, Thomas R, Siegmann B, Rossini M, Celesti M, Schütttemeyer D, Kraska T, Müller O, Rascher U. Unmanned aerial systems (UAS)-based methods for solar induced chlorophyll fluorescence (SIF) retrieval with non-imaging spectrometers: state of the art. Remote Sens 2020;12(10):1624. <https://doi.org/10.3390/rs12101624>
- [6] Choi W, Ryu Y, Kimm H, Belwalkar A, Poblete T, Yeon I, Ryu J-H, Lee K-D, Zarco-Tejada PJ. Detecting plant physiological changes under stress using sun-induced chlorophyll fluorescence from mid-spectral resolution imagery. Remote Sens Environ 2026;332:115085. <https://doi.org/10.1016/j.rse.2025.115085>
- [7] Kneer C, Burkart A, Bongartz J, Siegmann B, Bendig J, Jenal A, Rascher U. A snapshot imaging system for the measurement of solar-induced chlorophyll fluorescence—addressing the challenges of high-performance spectral imaging. IEEE Sens J 2023;23(19):23255–69. <https://doi.org/10.1109/JSEN.2023.3297054>
- [8] Plascyk JA, Gabriel FC. The fraunhofer line discriminator MKII—an airborne instrument for precise and standardized ecological luminescence measurement. IEEE Trans Instrum Meas 1975;24(4):306–13. <https://doi.org/10.1109/TIM.1975.4314448>
- [9] Agisoft. Agisoft metashape (version 2.2.1) (software) [version 2.2.1.20641]. <https://www.agisoft.com/downloads/installer/>.
- [10] Toffanin P, WebODM Contributors. WebODM: user-friendly, commercial-grade software for processing aerial imagery. <https://doi.org/10.5281/zenodo.3965771>. <https://github.com/WebODM/WebODM>.
- [11] Schönberger JL, Frahm J-M. Structure-from-motion revisited, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR);2016;pp. 4104–13, <https://doi.org/10.1109/CVPR.2016.445>

- [12] Pandey A, Pati UC. Image mosaicing: a deeper insight. *Image Vis Comput* 2019;89:236–57. <https://doi.org/10.1016/j.imavis.2019.07.002>
- [13] Li X, Feng R, Guan X, Shen H, Zhang L. Remote sensing image mosaicking: achievements and challenges. *IEEE Geosci Remote Sens Mag* 2019;7(4):8–22. <https://doi.org/10.1109/MGRS.2019.2921780>
- [14] Ferrer J, Elibol A, Delaunoy O, Gracias N, Garcia R. Large-area photo-mosaics using global alignment and navigation data. In: *OCEANS 2007*; 2007. p. 1–9. <https://doi.org/10.1109/OCEANS.2007.4449367>
- [15] Elibol A, Gracias N, Garcia R. Efficient topology estimation for large scale optical mapping. In: Vol. 82 of *Springer tracts in advanced robotics*. Springer; 2013. <https://doi.org/10.1007/978-3-642-30313-5>
- [16] Salattna S, Elibol A, Siegmann B, Kneer C, Scharr H, Rascher U, Bendig J. High spatial resolution imaging of solar-induced chlorophyll fluorescence (SIF) from an uncrewed aerial vehicle (UAV): development and evaluation of image-processing workflows. *IEEE J Sel Top Appl Earth Obs Remote Sens* 2026;1–19. <https://doi.org/10.1109/JSTARS.2026.3717082>
- [17] Fischler MA, Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun ACM* 1981;24(6):381–95.
- [18] Lowe DG. Distinctive image features from scale-invariant keypoints. *Int J Comput Vis* 2004;60:91–110. <https://doi.org/10.1023/B:VISI.0000029664.99615.94>
- [19] Buffat J, Bendig J, Elibol A, Scharr H, Choza Farias S, Salattna S. SIFMap: a Python toolbox for creating solar-induced fluorescence maps from UAV-Borne snapshot imaging data - test dataset; Jul 2026. <https://doi.org/10.26165/JUELICH-DATA/2KDXUL>
- [20] Hartley R, Zisserman A. *Multiple view geometry in computer vision*. 2nd ed. Harlow, UK: Cambridge University Press; 2004.
- [21] Ma Y, Soatto S, Kosecka J, Sastry SS. *An invitation to 3-D vision: from images to geometric models*. SpringerVerlag; 2003.