ICEPY4D: A PYTHON TOOLKIT FOR ADVANCED MULTI-EPOCH GLACIER MONITORING WITH DEEP-LEARNING PHOTOGRAMMETRY

F. Ioli [1,*], F. Barbieri [1], F. Gaspari [1], F. Nex [2], L. Pinto [1]

[1] Politecnico di Milano, Dep. of Civil and Environmental Engineering, Milan, Italy -
(francesco.ioli, federico2.barbieri, federica.gaspari, livio.pinto)@polimi.it
[2] Dept. of Earth Observation Science (EOS), Faculty ITC, University of Twente, Enschede,
The Netherlands - f.nex@utwente.nl

KEY WORDS: Short-term glacier monitoring, Time-lapse cameras, Wide baseline, SuperGlue, LOFTR, Structure-from-Motion, Open source

ABSTRACT:

Glacier monitoring plays a crucial role in understanding the impacts of climate change on these dynamic natural systems. One or more time-lapse cameras are often employed to acquire short-term observations of glacier flow dynamics. However, the lack of multi-camera photogrammetric software packages for multi-temporal 3D scene reconstruction, especially in case of wide camera baselines, hinders the application of Structure-from-Motion techniques to these scenarios. To address this, we present ICEpy4D, a novel Python toolkit designed for 4D monitoring of alpine glaciers using low-cost time-lapse cameras and state-of-the-art computer vision techniques. ICEpy4D leverages deep-learning-based matching algorithms to solve 3D reconstruction with wide camera baselines, making it well-suited for challenging scenarios encountered in mountainous regions. The toolkit offers comprehensive functionalities for multi-epoch monitoring, enabling short-term glacier 3D reconstruction and extraction of relevant information from time-series point clouds, such as volume variations and glacier retreat. In a pilot study on the Belvedere Glacier northern snout (Italian Alps), ICEpy4D estimated glacier volume loss of $63 \times 10^3\,\mathrm{m}^3$ of ice and $\sim 17.5\,\mathrm{m}$ of retreat. Results showcased the toolkit's potential for analyzing a glacier ice cliff, with prospects for application to other glaciers with varying characteristics. ICEpy4D is actively being developed as an open-source project at `github.com/labmgf-polimi/icepy4d/`, promoting ease of extension and customization.

1. INTRODUCTION

Time-lapse cameras are commonly used in remote alpine environments for qualitative and quantitative data collection, especially in monitoring glacier flow dynamics. They are cost-effective, provide images with high temporal frequencies (e.g., daily), and require minimal maintenance (Messerli and Grinsted, 2015). Digital Image Correlation (DIC) techniques are often employed to estimate in-plane velocities from sequences of monocular images and derive the glacier flow velocity (Giordan et al., 2016, Giordan et al., 2020). However, in most cases, only one camera is employed, which prevents the application of photogrammetry and Structure-from-Motion (SfM) for comprehensive 4D glacier monitoring.

Existing software packages such as ImGraft (Messerli and Grinsted, 2015), Pointcatcher (James et al., 2016) and PyTrx (How et al., 2020) allow for calculating glacier velocities from oblique time-lapse images using feature tracking. These solutions, however, are limited to monoscopic camera systems. Common photogrammetric software packages for 3D scene reconstruction from multi-camera systems, such as Agisoft Metashape or Colmap (Schönberger and Frahm, 2016) are more suited for solving SfM problem, with one or more moving cameras aimed at reconstructing a static scene, but they are not designed for multi-temporal processing as they rely on traditional Feature-Based Matching (FBM) techniques for matching on local features based on descriptors' similarity.

In mountainous areas, environmental constraints often restrict the number of feasible camera locations, forcing to install the cameras with wide baselines. As a consequence, matching corresponding points between images becomes a challenging task for traditional Feature-Based Matching (FBM) (Yao et al., 2021). In recent years, several Deep Learning (DL)-based feature matching algorithms have shown promising results in terms of performance in wide-baseline scenarios (Chen et al., 2021, Remondino et al., 2022). Some open-source projects for performing visual localization with DL matching techniques have been developed including HLOC (Sarlin et al., 2019) and GTSFM (Baid et al., 2021). Despite begin promising libraries, they are mostly limited to solving 6-degree-of-freedom problems and they are not designed for a multitemporal processing. To the best of the authors' knowledge, there are no available solutions for performing wide-baseline multi-temporal stereo or multi-camera photogrammetric reconstruction.

This work introduces ICEpy4D (Image-based toolbox for Continuous monitoring of glaciers' Evolution), a Python-based toolkit for multi-epoch glacier 3D reconstruction and monitoring. ICEpy4D bridges the gap between the geomorphological/glaciological and computer-vision communities by incorporating learning matching techniques for challenging 3D/4D scene reconstruction problems. It integrates state-of-the-art deep-learning-based matching for wide baselines and includes functionalities for processing and extracting relevant information from time-series of point clouds.

An application of ICEpy4D using low-cost stereo-cameras on Belvedere Glacier (Italian Alps) is presented as a pilot-study. However, the main focus of the paper is to describe the software package and its functionalities. A comprehensive presentation of the monitoring results from the Belvedere Glacier

* Corresponding author

will be covered in a separate paper. ICEpy4D is actively being developed as an open-source project at `github.com/labmgf-polimi/icepy4d/` (current version is v1.8.0).

2. FIELD SETUP AND DATA

ICEpy4D was tested on the Belvedere Glacier, an alpine glacier in Valle Anzasca, Italy, located at approximately 45° 58' N 7° 55' E. It spans from ∼2250 m a.s.l. to ∼1800 m a.s.l. and has two distinct lobes. The northern lobe features a sub-vertical ice cliff, the source of the River Anza, with exposed bare ice reaching ∼50 m in height.

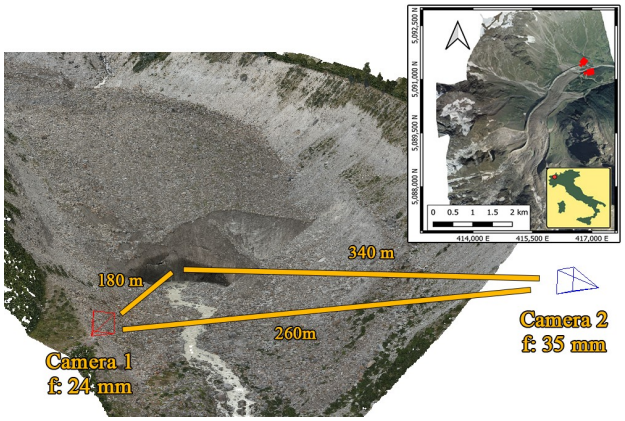


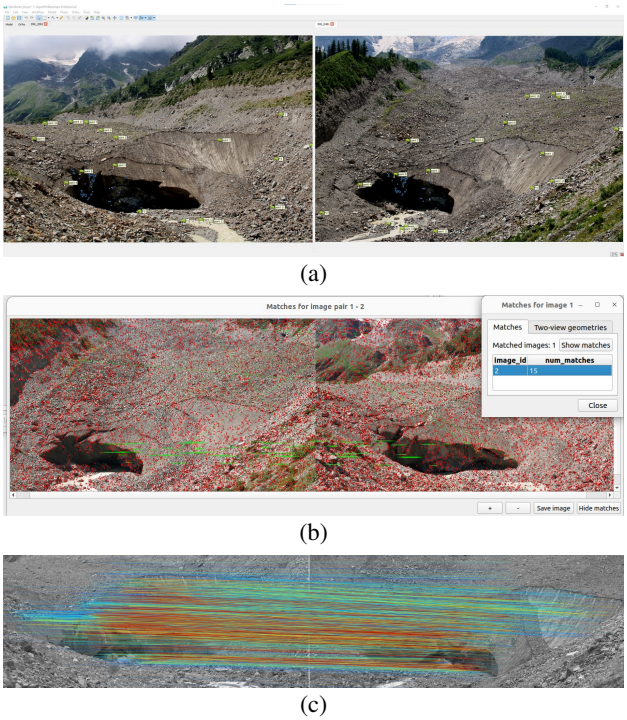
Figure 1. The Belvedere Glacier northern lobe used as study area and the camera setup.

The northern lobe of the Belvedere Glacier was chosen as the test site for ICEpy4D as part of an ongoing long-term monitoring project (Ioli et al., 2023b). Historical aerial images were used for photogrammetric modeling from 1977 to 2001 (De Gaetani et al., 2021), and since 2015, yearly in-situ UAV and GNSS surveys have been conducted to monitor glacier evolution (Ioli et al., 2021).

In summer 2021, a low-cost stereoscopic system with two handmade time-lapse cameras was installed on the northern lobe to observe sub-seasonal glacier dynamic. Each monitoring station consisted of a DSLR camera Canon Eos 1200D, an Arduino microcontroller for camera triggering and a Raspberry Pi Zero with a SIM card for remote image transfer. The system was described in detail in (Ioli et al., 2023a). The two monitoring stations were installed on opposite sides of the northern lobe moraines, with one camera placed ∼180 m from the terminal ice cliff and the other ∼340 m away, resulting in a baseline of ∼260 m. To maintain a comparable ground sample distance of ∼3.5 cm px^{-1}, lenses with different focal lengths (24 mm and 35 mm) were used. The cameras were set up on topographic tripods at a height of ∼2 m above the ground (Ioli et al., 2023a). Image acquisition occurred remotely through a web-interface, capturing two images per day around noon, at 12:00 and 13:00 UTC.

3. MOTIVATION

The Belvedere camera setup posed a challenge for traditional hand-crafted matching techniques due to the camera baseline comparable to the average camera-object distance. These techniques struggled to find enough corresponding points with wide-baseline matching. Even popular photogrammetric software packages like Metashape, Pix4D Mapper and the open-source solution COLMAP (Schönberger and Frahm, 2016), which rely on SIFT (Lowe, 2004) or faster variants like RootSIFT (Arandjelović and Zisserman, 2012) for feature matching, provided limited and unevenly distributed tie points. Additionally, these few tie points were primarily concentrated in the central part of the ice cliff (Fig. 2). Similar results were observed when using other traditional detectors and descriptors like SURF, KAZE and ORB, along with matching algorithms from the OpenCV library. The lack of tie points found with existing SfM software packages hindered the estimation of camera poses.



(a)

(b)

(c)

Figure 2. Comparison with feature matching by using different approaches: (a) Agisoft Metashape (blue points are the matched points); (b) COLMAP (green lines); (c) SuperPoint+SuperGlue (colored lines). The color of the SuperGlue matches represents the matching score, with colors ranging from blue to red.

Recently, DL-based feature matching algorithms have shown promising performance in challenging scenarios (Yao et al., 2021, Chen et al., 2021, Remondino et al., 2022). For example, SuperGlue, an end-to-end Convolutional Neural Network (CNN) for feature extraction and matching, has proven effective in various real-world scenarios, including low-quality images from webcams (Wu et al., 2021), historical images (Maiwald et al., 2021) and images with different viewpoints or acquisition conditions (Bellavia et al., 2022). Another popular detector-free end-to-end matcher gaining traction is LOFTR (Sun et al., 2021), which is also finding applications in the computer vision and remote sensing communities (Bellavia et al., 2022, Liang et al., 2023). ICEpy4D implements both SuperGlue and LOFTR for feature matching, allowing for solving challenging 3D/4D scene reconstruction problems with wide camera baselines, which were not feasible with traditional FBM techniques.

4. ICEPY4D

ICEpy4D is organized in modules for solving different steps of scene reconstruction and point cloud analysis problem. The main workflow (Fig. 3) is composed of different steps: (i) finding corresponding features between stereo-pairs; (ii) tracking features on single-camera time-series with template matching e.g., for automatically detect targets on monoscopic image sequences; (iii) solving relative and absolute orientation of stereo/multi cameras; (iv) solving a Bundle Adjustment (BA) using external libraries (e.g., Agisoft Metashape Python API or COLMAP), including Ground Control Points (GCPs) and self-calibration support; (v) building dense reconstruction and meshing with external libraries; (vi) processing a time-series of point clouds to estimate volume differences and glacier retreat. A list of the main classes and modules of ICEpy4D is presented in Tab. 1

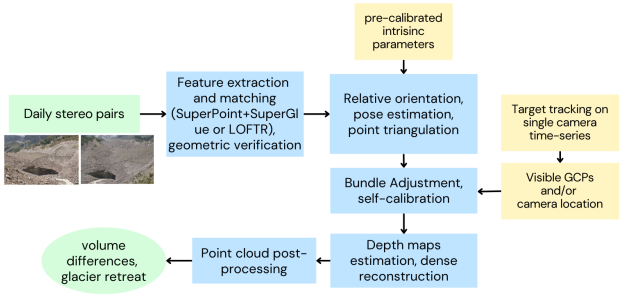


Figure 3. Overview of the ICEpy4D workflow.

4.1 Data preparation

To perform stereo or multi-camera 3D reconstruction with ICEpy4D, we used a checkerboard calibration target (Zhang, 2000) to pre-calibrate the cameras. This calibration allowed us to estimate intrinsic parameters such as the principal distance, principal point, and radial and tangential distortion coefficients, following the *Full-OPENCV* camera model, similar to COLMAP (Schönberger and Frahm, 2016). The estimated interior orientation parameters need to be saved in separate text files for each camera used and serves as the initial calibration for BA, where the interior orientation can be refined by self-calibration.

First, we converted all acquired RAW images to JPEG format using RawTherapee, with a compression quality of 95 %. The converted images were manually inspected to exclude unsuitable ones for stereo reconstruction due to issues like rain or low clouds. We organized the selected images into separate folders for each camera.

For automatic processing of the time-series, the image and world coordinates of the available GCPs on each image must be stored in separate text files with the same name as the corresponding image file. GCPs can be manually detected on all images or automatically tracked using the template matching routine (Sec. 4.2).

The ICEpy4D processing is initialized by creating an empty Epochs object, which is filled by adding Epoch objects at each iteration of the process.

4.2 Target tracking

To account for small camera rotations that can be caused by adverse meteorological conditions in the mountain environment (especially wind guts), we implemented a tracking routine in ICEpy4D. This routine, based on the ImGraft TemplateMatch function (Messerli and Grinsted, 2015), tracks the same targets on all images in the time-series. It allows us to detect a template defined on a reference image on a target image through cross-correlation.

In ICEpy4D, the template matching routine uses the orientation correlation algorithm (Fitch et al., 2002), which is more robust against illumination changes compared to normalized cross-correlation (Heid and Kääb, 2012). The template matching routine estimates the image coordinates of the GCPs on all the images in the time-series and exports them to a text file. This file serves as input for the subsequent processing steps in ICEpy4D.

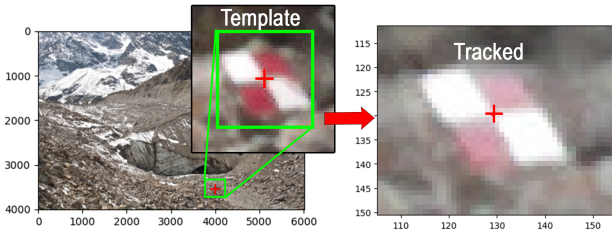


Figure 4. Example of template matching used to track GCPs on a monocular image sequence. The green square represents the template on the reference image that is searched in all the other images of the sequence. The red cross marks the estimated position of the center of the template in a new image.

4.3 Feature Matching

For matching corresponding points between stereo-pairs, ICEpy4D implements both the combination of SuperPoint (DeTone et al., 2018) and SuperGlue (Sarlin et al., 2020) and LOFTR (Sun et al., 2021). SuperPoint is a CNN that detects interest points and extract their corresponding 256-values descriptors. SuperGlue matches the extracted keypoints using an attentional graph neural network, considering both descriptor similarity and pixel positions. On the other hand, LOFTR establishes dense pixel-wise matches at a coarse level and then refines good matches at a fine level. It employs self and cross attention layers in a Transformer model to obtain feature descriptors conditioned on both images. LOFTR was implemented in the well-known KORNIA library (Riba et al., 2020), a state-of-the-art differentiable computer vision library based on PyTorch.

To perform the matching, we utilized pre-trained models for both SuperPoint/SuperGlue and LOFTR. The SuperGlue model was trained on the MegaDepth dataset (Li and Snavely, 2018), which includes various outdoor scenes and makes SuperGlue particularly effective in matching corner-like features detected by SuperPoint. We chose not to fine-tune the SuperGlue and LOFTR models to test the replicability of the matching solutions in different scenarios without the need for a dedicated ground truth dataset. Acquiring such a dataset, especially for remote mountain environments, in fact, can be challenging. For a future software release, we plan to acquire or gather a substantial dataset of UAV and ground-based images capturing various glaciers, including debris-covered and bare-ice glaciers. This dataset will enable us to fine-tune matching models specifically for glacier reconstructions, offering broader applicability beyond the Belvedere glacier case study.

Name	Brief description
classes (M)	Module that contains all the main classes of ICEpy4D.
matching (M)	Module that integrates functionalities for feature detection and matching by using CNN algorithms.
metashape (M)	Module for interfacing ICEpy4D with Agisoft Metashape.
post_processing (M)	Module that implements functionalities post-processing series of stereo point clouds to derive volume variations, glacier retreat and other information.
sfm (M)	Module that implements functionalities for estimating relative and absolute orientation of two cameras, calculating the exterior orientation of a single camera based on GCPs (i.e., space resection), and triangulating points.
Epoch (C)	Class that acts as a container for storing all the variables related to a single epoch: epoch timestamp, images, camera, targets, features, points. Epoch class integrates functions for saving and reading data to file.
Epochs (C)	Class for storing a sequence of Epoch objects for multitemporal processing. Each Epoch is identified by its timestamp and by an unique progressive integer identifier.
Image (C)	Class for managing images and metadata. It allows for reading timestamps from exif data.
Camera (C)	Class for storing interior (intrinsics matrix and distortions) and exterior orientation (camera pose) of each camera. It provides methods for projecting 3D points on the image plane. A Camera object is initialized by reading a calibration file or by estimating the camera focal length from image EXIF data.
Calibration (C)	Class for reading a calibration file with different formats (e.g. OpenCV, Agisoft) and building a Camera object.
Feature (C)	Class for storing a 2D features on images. A Feature is composed of a unique *track_id*, the xy image coordinates of the keypoints, and optionally, the descriptor and the matching score.
Features (C)	Class for storing a series of Feature objects. Each Feature is indexed by a unique *track_id*. A Features object contains a dictionary of *key-value* pairs, with the structure *track_id: Feature*.
Point (C)	Class for storing a 3D point in object space.
Points (C)	Class for storing a list of Point objects. Each Point is indexed with a unique *track_id* that links it to the corresponding Feature on the images. Points class has methods for converting points to a Numpy array, as well as adding points from a Numpy array and filtering points by boolean masks.
PointCloud (C)	Class for storing 3D points as an Open3D point cloud object, from which it is possible to access all the Open3D functionalities for managing point clouds.
Target (C)	Class for storing the image and world coordinates of the GCPs. A Target object is initialized by reading a text file containing the coordinates of the GCPs.
Tiler (C)	Class belonging to the *matching* module for subdividing images in regular tiles.
ImageMatcherBase (C)	Base class that implement the interface and some basic functions for feature matching with a generic algorithm. This class must be subclassed to implement the actual feature matching.
SuperGlueMatcher (C)	Subclass of *ImageMatcherBase* belonging to the *matching* module for performing feature matching by using the SuperGlue CNN.
LOFTRMatcher (C)	Subclass of *ImageMatcherBase* belonging to the *matching* module for performing feature matching by using the LOFTR, implemented in KORNIA library.
RelativeOrientation (C)	Class of the *sfm* module for solving relative orientation of two cameras.
Triangulate (C)	Class of the *sfm* module triangulating corresponding points in the object space, given two Camera objects with known poses and two Features objects.
AbsoluteOrientation (C)	Class of the *sfm* module for solving absolute orientation of Cameras and Points.
MetashapeProject (C)	Class of the *metashape* module for building a Metashape project by using Metashape Python API. It allows for importing ICEpy4D solutions, solving BA and dense reconstruction.
TrackTargets (C)	Class for tracking the position of a point (usually a GCP target) on a sequence of monocular images.
DemOfDifference (C)	Class of the *post_processing* module for computing the DEM of Difference between two point clouds.

Table 1. Description of main classes (labelled as C) and modules (labelled as M) of ICEpy4D. For additional information, one can refer to the documentation accessible from the ICEpy4D repository.

Feature matching is conducted using the *SuperGlueMatcher* and *LOFTRMatcher* classes, implemented within the *matching* module. The two classes have the same interface so that they can be used interchangeably. This matching process is implemented entirely with PyTorch, allowing for GPU acceleration to enhance performance.

The user can specify if the matching is performed on full-resolution images, on downsampled images or upsampled images. This is controlled by the *Quality* parameter, where *Quality.HIGH* means performing the matching on full resolution images, *Quality.HIGHEST* means upsampling the images by a factor two, and *Quality.MEDIUM* and *Quality.LOW* means downsampling the images by a factor two and four respectively.

To guarantee the highest collimation accuracy, by default the

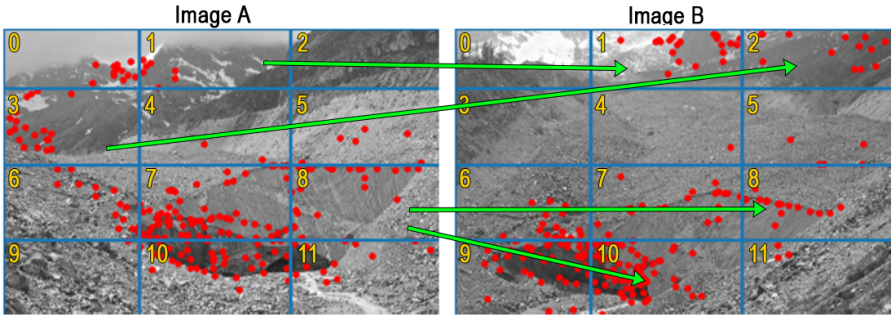


Figure 5. Feature matching by*tile preselection*: a first matching is carried out on downsampled image A and B (red dots). For a second matching step with full resolution images, corresponding tiles are selected based on the location of the matched keypoints found on the downsampled images (e.g., tile A1 is matched with B1 and A3 with B2). A tile on image A can also be matched with multiple tiles on image B (e.g., A8 is matched with both B8 and B10). Tiles with a low number of matching (e.g., A4 and B5) are discarded.

matching is performed on full resolution images. However, due to limited memory capacity in mid-class GPUs, high-resolution images captured by DSLR cameras may not fit into GPU memory. To overcome this limitation, ICEPy4D divides the images into smaller regular tiles with maximum dimension of 2000 px, computed over a regular grid. The matching can be executed in either *exhaustive* or *preselection* mode. In the *exhaustive* mode, all tiles are matched with all tiles from the other camera, whereas the *preselection* mode only matches tiles that are likely to contain corresponding points. In the *preselection* mode, the matching process consists of two steps (Fig. 5). First, a matching is performed on downsampled images. Subsequently, the full-resolution images are subdivided into regular tiles, and only the tiles that have corresponding features in the low-resolution images are selected as candidates for a second matching step. The selected tiles are matched using the same procedure as before. To ensure keypoints near the tile boundaries are detected, tiles can be enlarged with a buffer area in each direction to achieve overlapping. The features matched within each tile are then reassembled to recover their image coordinates in the original image for geometric verification.

Geometric verification of the matches is performed by using Pydegensac (Mishkin et al., 2015), that allows for robustly estimate the fundamental matrix. The maximum re-projection error to accept a match is set to 1.5 px by default, but it can be changed by the user. The successfully matched features, together with their descriptors and scores, are saved as a Features object for each camera and stored into the current Epoch object.

4.4 Relative and Absolute Orientation

ICEpy4D implements a class for estimating the relative orientation of two or more cameras based on the matched features. This step is necessary because it allows for deriving an approximate solution that can be refined during the BA, carried out with external software packages.

Relative and absolute orientation are implemented in specific classes within the *sfm* module and they are based on OpenCV library. The *RelativeOrientation* class allows for estimating the relative pose of a *new* camera with respect to a *reference* one (e.g., the first one). Estimated relative rotation and translation are stored into the new Camera object. If the reference camera has already been oriented with respect to an external reference system, the *RelativeOrientation* class allows for updating the camera pose of the new camera with respect to the external reference system. Additionally, if the camera baseline is known, the relative translation is scaled correspondingly. Given two (or more) camera objects with estimated pose, the features matched as described in Sec. 4.3 are triangulated in the object space by using the *Triangulate* class, available in the *sfm* module.

Absolute orientation is performed by using the *AbsoluteOrientation* class, which allows to estimate a Helmert transformations between two different reference systems, e.g., the local reference system defined by the first camera and a world (local or global) reference system. To this end, at least three GCPs must be available to estimate the transformation matrix. The two camera locations, if known, can be used as additional GCPs.

If more than two cameras are available, the relative orientation is performed sequentially, by estimating relative orientation between two cameras at a time. In a future release of ICEpy4D, a preselection of the images to be matched will be carried out, e.g., by place recognition algorithms such as NetVLAD (Arandjelović et al., 2016).

4.5 Bundle Adjustment

BA is performed with external software packages, such as Agisoft Metashape or COLMAP. ICEpy4D allows for interfacing with these software packages by using the *MetashapeProject* class (based on the Agisoft Metashape Python API) and the open-source library PyColmap (`https://github.com/colmap/pycolmap`).

Although it is a commercial and closed-source software package, Agisoft Metashape is currently the preferred solution as it allows for performing a full bundle adjustment, including GCPs, refining cameras' interior orientation by self-calibration and giving weights to the observations based on their variance. However, using Agisoft Metashape API requires a valid license of the software. This is clearly a limitation, but Agisoft Metashape is currently widespread in the photogrammetric community, as it is one of the most complete SfM software packages available. Therefore, we believe that is worthing to include it in ICEpy4D.

In particularly, the *MetashapeProject* class allows for building a Metashape project by importing the images and the matched features, setting the cameras' interior orientation and fix the desired parameter to not be optimized during the BA. The *MetashapeProject* class also allows for running Metashape BA, computing depth maps and running dense reconstruction. Camera and Points objects of the current epoch are then updated with the refined parameters.

In the case of the Belvedere Glacier, we used Metashape to perform the BA, as at least 2 GCPs were always visible on the images. This allowed us to perform a self-calibration of the cameras for refining the estimation of the cameras' principal distance. In fact, it is known that the camera principal distance is subjected to temperature variations, which can be relevant in the case of cameras mounted in mountain environments.

Alternatively to Metashape, the BA can be performed with COLMAP, by using the *incremental_mapping* function, integrated in PyColmap. This allows to refine the cameras' exterior orientation, the points' coordinates and the camera interior orientation. However, COLMAP does not support the use of GCPs directly in the BA, but they can only be used to perform an abolute orientation a-posteriori (i.e., with a Helmert transformation). This hinders the possibility of performing a self-calibration when only two cameras, with different interior orientation, are available, as in the case of the Belvedere Glacier.

4.6 Dense Reconstruction

Similarly as for the BA, the dense reconstruction is performed with external software packages Metashape or COLMAP, depending on which software was chosen for the BA. Both the software packages implement semi-global matching algorithms (Hirschmüller et al., 2012).

In the case of the Belvedere Glacier, dense reconstruction of the ice cliff is carried out by Agisoft Metashape. In fact, although Agisoft Metashape was unable to perform feature matching with wide baselines and estimate the camera poses, as it employes hand-crafted features, but it was effective in performing dense matching by applying semi-global matching algorithms, if the camera poses are known. Depth maps are built from full-resolution images (i.e., *highest quality* parameter in Agisoft Metashape) with *mild filtering*. As Agisoft Metashape is a closed-source software package, no additional information about the dense matching algorithms is available. Estimated depth maps are used to reconstruct a dense point cloud and a triangulated mesh.

4.7 Point Cloud Processing

ICEPy4D integrates a module for processing a time-series of point clouds, in particular to extract information about volume variations and glacier retreat. The point cloud processing is performed with the open-source library Open3D (Zhou et al., 2018) and CloudComPy, which is the Python API for CloudCompare.

The daily variation in ice volume is determined by calculating the DEM of Differences (DOD) in the streamwise direction from a pair of point clouds. To this end, the point clouds are first converted to triangulated meshes using the Poisson meshing algorithm (Kazhdan et al., 2006) in Open3D library to fill small gaps. The meshes are cropped to the same bounding polygon and uniformly sampled for a homogenous point density. The sampled point clouds are rasterized to a vertical YZ plane (Fig. 6) with a grid step of $0.3\,\mathrm{m}$, maintaining the same origin and extent. The volume variation between two epochs is calculated by differentiating the rasters on a cell-by-cell basis (Fig.6). To account for incomplete coverage and holes in the point clouds (e.g., due to snow or shadows) that couldn't be filled by mesh sampling techniques, the percentage of full cells in the raster is calculated. The raw volume variation estimate is normalized by the percentage of full cells in both raster. This normalization compensates for empty cells in one of the point clouds, which would underestimate the resulting volume.

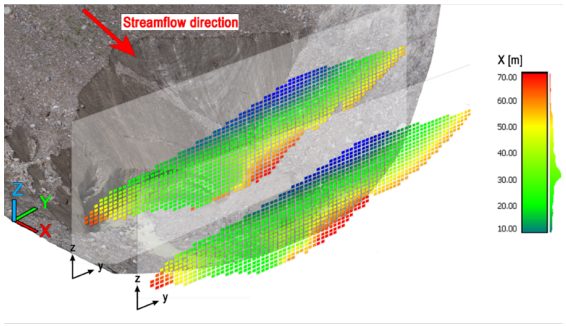


Figure 6. Scheme of Dem of Difference (DOD) approach used to compute volume variations at the glacier terminal ice cliff.

In addition to DOD computation, most of the CloudCompare functionalities, including M3C2 cloud-to-cloud distance computation (Lague et al., 2013), are exposed to ICEpy4D by means of the CloudComPy library. Similarly, through Open3D library, ICEpy4D allows for applying various filters to the point clouds, such as voxel downsampling, statistical outlier removal, radius outlier removal, as well as for computing the normal vectors of the points or filtering the point cloud based on a polyline.

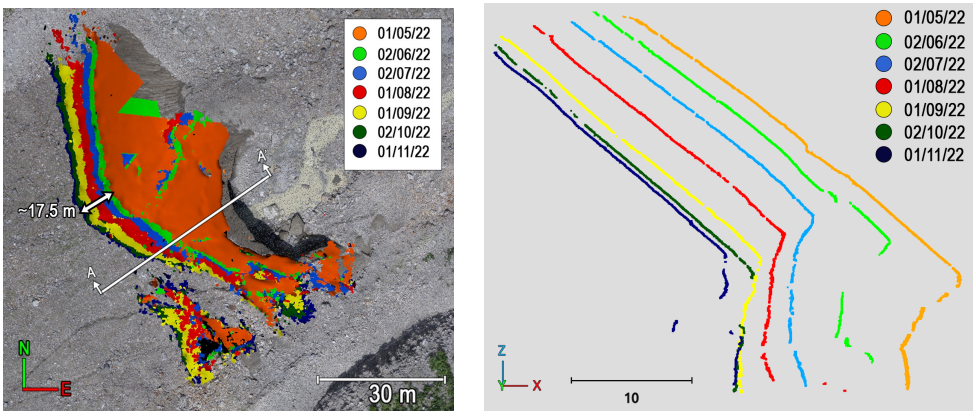
5. THE BELVEDERE GLACIER CASE STUDY

For the Belvedere case study, we used a set of 158 pairs of stereo images acquired from 01/05/2022 to 13/11/2022, during the snow-free season. The presence of snow, in fact, clearly affects the quality of the stereo reconstruction and made it difficult (or even impossible) to locate the GCPs on the images.

Feature matching was carried out with SuperPoint and SuperGlue. A number ranging from 1000 to 3500 valid matches was obtained for each pair of images, depending mostly on weather conditions. At each epoch, the approximate solution was derived by estimating the relative and absolute orientation of the cameras, followed by a BA carried out with Metashape.

As the cameras may slightly rotate due to wind, the location of the two camera was supposed as fixed, while the camera rotations with respect to the World Reference System (WRS) were estimated at each epoch during BA. Moreover, to ensure WRS coherence between different epochs and refine cameras' principal distance, at least three GCPs located on stable areas in front of the ice cliff and along the glacier moraines were manually detected at the first epoch and tracked on all the others. GCPs were introduced in the BA, properly weighted by their variance. After the BA, on average, the reprojection error of all the features on the two images at each epoch was 0.45 pixels, with a standard deviation of approximately 0.3 pixels. The dense reconstrucion was carried out with Metashape, using full resolution images to compute depth maps, from which dense point clouds and meshes were derived for every epoch.

An assessment of the accuracy of the dense reconstruction was carried out by computing cloud-to-cloud distances with M3C2 algorithm (Lague et al., 2013) between the stereo point cloud and a reference photogrammetric point cloud derived from a UAV flight performed on 28/07/2022. The results of the assessment were presented by Ioli et al. (Ioli et al., 2023a) and showed



(a) (b)

Figure 7. (a) Monthly point clouds built with ICEpy4D from May to November 2022. (b) Vertical cross sections AA' of the point clouds extracted at the location marked in (a).

a mean of the distances of $0.01\,\mathrm{m}$ and a standard deviation of $0.08\,\mathrm{m}$, which is less than 3 times the image GSD of $3.5\,\mathrm{cm}$.

From the daily series of point cloud, one point cloud at the beginning of each month was selected for visualize the glacier retreat (Fig. 7a). The retreat was estimated by M3C2 algorithm as $\sim$17.5 m from May to November 2022. Ice volume loss was derived by DOD from pairs of point clouds spaced by 5 days of time interval in order to increase the signal-to-noise ratio (based on the rate of movement and accuracy of the estimated dense point cloud). From May to November 2022, $63 \times 10^3\,\mathrm{m}^3$ of ice was lost (Fig. 8).

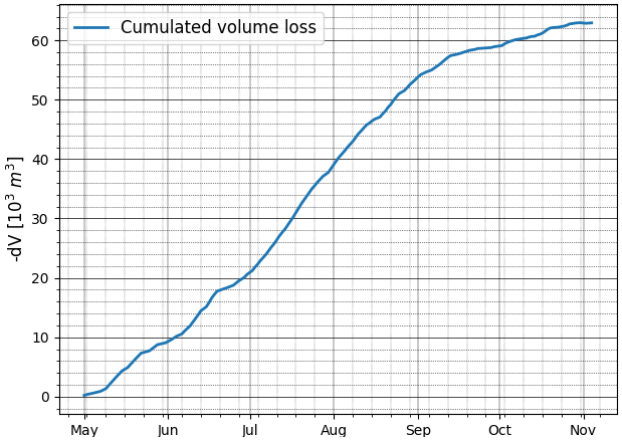


Figure 8. Cumulated volume variations from May to November 2022, estimated by Dem of Differences of stereo point clouds, spaced by an interval of 5 days.

6. CONCLUSIONS

In this paper, we present ICEpy4D, a novel Python-based toolkit designed for 4D monitoring of alpine glaciers using low-cost setups, such as time-lapse cameras composed of off-the-shelf components. ICEpy4D leverages state-of-the-art CNN matching algorithms to solve 3D reconstruction with wide baselines, making it suitable for glacier monitoring.

The software was successfully tested on the Belvedere Glacier, allowing for the derivation of daily volume variations at the glacier snout and glacier retreat. Although the glacier's debris cover and dirty ice terminal cliff provided favorable conditions for 3D reconstruction, due to the presence of distinct patterns in the images, we believe the method is applicable to other glaciers with different characteristics Previous studies have shown successful 3D reconstructions of debris-free glaciers using UAV and ground based SfM with traditional feature matching techniques (Gindraux et al., 2017, Belloni et al., 2023, Taylor et al., 2023) It is worth noting that these examples utilized traditional feature matching techniques. With state-of-the-art DL sparse and dense matching techniques, results can be further improved.

Being written in Python and modular, ICEpy4D offers ease of extension and customization for future improvements. Future plans for ICEpy4D include interfacing with libraries like py4dgeo (Anders et al., 2021) to enable multitemporal processing of point clouds. Moreover, CERES library will be tested as an open-source replacement for Metashape, incorporating GCPs and self-calibration in BA.

REFERENCES

Anders, K., Winiwarter, L., Mara, H., Lindenbergh, R., Vos, S. E., Höfle, B., 2021. Fully automatic spatiotemporal segmentation of 3D LiDAR time series for the extraction of natural surface changes. *ISPRS J. Photogramm. Remote Sens*, 173, 297-308.

Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J., 2016. NetVLAD: CNN architecture for weakly supervised place recognition. *Proc. CVPR IEEE*.

Arandjelović, R., Zisserman, A., 2012. Three things everyone should know to improve object retrieval. *2012 Proc. CVPR IEEE*, IEEE, 2911–2918.

Baid, A., Driver, T., Jiang, F., Krishnan, A., Lambert, J., Liu, R., Singh, A., Upadhyay, N., Venkataramanan, A., Warrier, S., Womack, J., Wu, J., Wu, X., Dellaert, F., 2021. GTSFM: Georgia tech structure from motion. `https://github.com/borglab/gtsfm`.

Bellavia, F., Colombo, C., Morelli, L., Remondino, F., 2022. Challenges in image matching for cultural heritage: An overview and perspective. *Image Analysis and Processing. ICIAP 2022 Workshops*, Springer International Publishing, Cham, 210–222.

Belloni, V., Rita, M. D., Fugazza, D., Traversa, G., Hanson, K., Diolaiuti, G., Crespi, M., 2023. High-resolution high-accuracy orthophoto map and digital surface model of Forni Glacier tongue (Central Italian Alps) from UAV photogrammetry. *Journal of Maps*, 19(1), 2217508.

Chen, L., Rottensteiner, F., Heipke, C., 2021. Feature detection and description for image matching: from hand-crafted design to deep learning. *Geo-spat. Inf. Sci.*, 24(1), 58-74.

De Gaetani, C. I., Ioli, F., Pinto, L., 2021. Aerial and UAV Images for Photogrammetric Analysis of Belvedere Glacier Evolution in the Period 1977–2019. *Remote Sens.*, 13(18), 3787.

DeTone, D., Malisiewicz, T., Rabinovich, A., 2018. Superpoint: Self-supervised interest point detection and description. *Proc. CVPR IEEE*.

Fitch, A., Kadyrov, A., Christmas, W. J., Kittler, J., 2002. Orientation correlation. *BMVC*, Citeseer, 1–10.

Gindraux, S., Boesch, R., Farinotti, D., 2017. Accuracy Assessment of Digital Surface Models from Unmanned Aerial Vehicles' Imagery on Glaciers. *Remote Sensing*, 9(2).

Giordan, D., Allasia, P., Dematteis, N., Dell'Anese, F., Vagliasindi, M., Motta, E., 2016. A low-cost optical remote sensing application for glacier deformation monitoring in an alpine environment. *Sensors*, 16(10), 1750.

Giordan, D., Dematteis, N., Troilo, F., Segor, V., Godone, D., 2020. Close-range sensing of alpine glaciers. M. Kanao, D. Godone, N. Dematteis (eds), *Glaciers and the Polar Environment*, IntechOpen, Rijeka, chapter 7.

Heid, T., Kääb, A., 2012. Evaluation of existing image matching methods for deriving glacier surface displacements globally from optical satellite imagery. *Remote Sens. Environ.*, 118, 339-355.

Hirschmüller, H., Buder, M., Ernst, I., 2012. Memory Efficient Semi-Global Matching. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.*, I-3, 371–376.

How, P., Hulton, N. R. J., Buie, L., Benn, D. I., 2020. PyTrx: A Python-Based Monoscopic Terrestrial Photogrammetry Toolset for Glaciology. *Frontiers Earth Scie.*, 8.

Ioli, F., Bianchi, A., Cina, A., De Michele, C., Maschio, P., Passoni, D., Pinto, L., 2021. Mid-Term Monitoring of Glacier's Variations with UAVs: The Example of the Belvedere Glacier. *Remote Sens.*, 14(1), 28.

Ioli, F., Bruno, E., Calzolari, D., Galbiati, M., Mannocchi, A., Manzoni, P., Martini, M., Bianchi, A., Cina, A., De Michele, C., Pinto, L., 2023a. A Replicable Open-Source Multi-Camera System for Low-Cost 4D Glacier Monitoring. *Int. Arch. Photogramm.*, XLVIII-M-1-2023, 137–144.

Ioli, F., De Gaetani, C., Barbieri, F., Gaspari, F., Pinto, L., Rossi, L., 2023b. Belvedere glacier long-term monitoring open data. Zenodo. https://doi.org/10.5281/zenodo.7842348.

James, M. R., How, P., Wynn, P. M., 2016. Pointcatcher software: analysis of glacial time-lapse photography and integration with multitemporal digital elevation models. *J. Glaciol.*, 62(231), 159–169.

Kazhdan, M. M., Bolitho, M., Hoppe, H., 2006. Poisson surface reconstruction. *Symp. Geom. Process.*

Lague, D., Brodu, N., Leroux, J., 2013. Accurate 3D comparison of complex topography with terrestrial laser scanner: Application to the Rangitikei canyon (NZ). *ISPRS J. Photogramm. Remote Sens.*, 82, 10–26.

Li, Z., Snavely, N., 2018. Megadepth: Learning single-view depth prediction from internet photos. *Proc. CVPR IEEE*, 2041–2050.

Liang, C., Dong, Y., Zhao, C., Sun, Z., 2023. A Coarse-to-Fine Feature Match Network Using Transformers for Remote Sensing Image Registration. *Remote Sens.*, 15(13).

Lowe, D. G., 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2), 91–110.

Maiwald, F., Lehmann, C., Lazariv, T., 2021. Fully Automated Pose Estimation of Historical Images in the Context of 4D Geographic Information Systems Utilizing Machine Learning Methods. *ISPRS Int. of Geo-Inf.*, 10(11).

Messerli, A., Grinsted, A., 2015. Image georectification and feature tracking toolbox: ImGRAFT. *Geosci. Intrum. Meth.*, 4(1), 23–34.

Mishkin, D., Matas, J., Perdoch, M., 2015. MODS: Fast and robust method for two-view matching. *Computer Vision and Image Understanding*, 141, 81-93.

Remondino, F., Morelli, L., Stathopoulou, E., Elhashash, M., Qin, R., 2022. Aerial Triangulation with Learning-Based Tie Points. *Int. Arch. Photogramm.*, XLIII-B2-2022, 77–84.

Riba, E., Mishkin, D., Ponsa, D., Rublee, E., Bradski, G., 2020. Kornia: an open source differentiable computer vision library for pytorch. *Winter Conf. Appl. of Comput. Vis.*

Sarlin, P.-E., Cadena, C., Siegwart, R., Dymczyk, M., 2019. From coarse to fine: Robust hierarchical localization at large scale. *Proc. CVPR IEEE*.

Sarlin, P.-E., DeTone, D., Malisiewicz, T., Rabinovich, A., 2020. SuperGlue: Learning feature matching with graph neural networks. *Proc. CVPR IEEE*.

Schönberger, J. L., Frahm, J.-M., 2016. Structure-from-motion revisited. *Proc. CVPR IEEE*.

Sun, J., Shen, Z., Wang, Y., Bao, H., Zhou, X., 2021. LoFTR: Detector-free local feature matching with transformers. *Proc. CVPR IEEE*.

Taylor, L. S., Quincey, D. J., Smith, M. W., 2023. Evaluation of low-cost Raspberry Pi sensors for structure-from-motion reconstructions of glacier calving fronts. *Nat. Hazards Earth Sys.*, 23(1), 329–341.

Wu, T., Schindler, K., Albl, C., 2021. 3d reconstruction from public webcams. *Proc. CVPR IEEE*.

Yao, G., Yilmaz, A., Meng, F., Zhang, L., 2021. Review of Wide-Baseline Stereo Image Matching Based on Deep Learning. *Remote Sens.*, 13(16), 3247.

Zhang, Z., 2000. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Machine Intell.*, 22(11), 1330–1334.

Zhou, Q.-Y., Park, J., Koltun, V., 2018. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847*.