Using Micmac for generating Digital Elevation Model by processing pictures acquired from micro-UAV

J.-M Friedt[1], É. Bernard[2], F. Tolle[2]
[1] FEMTO-ST Institute, Time & Frequency, Besançon [2] ThéMA, Besançon

In the context of investigating the evolution of an Arctic glacier moraine, a commercial off-the-shelf micro-unmanned aerial vehicle (UAV) is used to collect azimuthal-view images over the moraine area. Digital elevation models and orthophotos are generated by assembling and geometrically correcting from topographic features using the Micmac software. The products of such signal processing steps are analyzed to estimate the resolution and accuracy of the measurements. A landslide is monitored between two dataset acquisition separated by one week.

Keywords: Micmac, QGis, UAV, digital elevation model (DEM), orthophoto

We have previously presented [1] the Micmac software [2] developed mainly by the MATIS laboratory hosted by the French National Geographic Institute (IGN). In this article, we aim at exploiting Micmac in the practical context of mapping the moraine of an Arctic glacier (Spitsbergen, Norway). Such a contextual presentation might appear trivial but is actually important: the area is not covered by vegetation (hence little terrain variation between two data acquisition as might be induced by plant motion due to wind), exhibit fast dynamics with erosion by flowing water, and free from most dangers met in urban areas (but nevertheless requires a flight authorization). We tackle the ability to monitor this area using a micro-UAV commercially available for less than 2500 euros. Indeed, while the relatively smooth glacier surface is readily mapped thanks to a few well selected measurement points and between which interpolation is reasonable, a moraine exhibit strong topographic variations between which interpolation might not be valid.

On the other hand, working in such an isolated area introduces new challenges [3]: reduced battery life expectancy at low temperatures and carrying the equipment on backpacks, which made us adopt micro-UAV when compared to the reference approaches we consider to be ground based or airborne LiDAR (Light Detection and Ranging [4] – the optical version of the RADAR), whose cost definitely defies most amateur work. This report does not aim at analyzing and interpreting geomorphological evolution patterns of the observed area, but to describe a processing flow of the acquired data and assess the resolution and accuracy of the digital elevation models (DEM) obtained for various dates separated by up to a week.

1 Data acquisition and expected resolution

Considering picture acquisition conditions prior to flying is mandatory. We define during this analysis the targeted resolution of features on the ground, overlap between images along the path and from one path to another for photogrammetric processing, and horizontal speed of the UAV while pictures are acquired over the area under investigation. We have bought and exploited a commercial DJI Phantom3 Professional: this model, new when acquired around Fall 2015, did not offer at the time automated flight planning, and all controls during flights were manually remote-controlled within visual line of sight and checking on the telemetry the GPS path over a background map as well as real time transmission of pictures acquired by the UAV (Fig. 1). While these options appeared initially as secondary gadgets with little scientific purpose, they appeared on the field most useful to make sure overlap between pictures along the flight path and between paths was sufficient. Micmac operates best when adjacent pictures overlap by 80%. Considering an angular field of view of 94° along the width of the picture, a flight elevation of 100 m induces a pixel width of 4 cm, hence below the decimeter we aim for but most significantly, meeting the regulation requirement of flying below 120 m (Fig. 2, right). Hence, the 2250 pixels along the picture height cover a ground span of 56 m. Considering a maximum horizontal speed of 10 m/s (which we use to map as much of the moraine area as possible during each flight, despite the risk of blurred images under low lighting conditions), grabbing one picture every second meets the overlap requirement. The control software provided by DJI does not allow acquiring pictures so close in time: we manually trigger the picture acquisition at time intervals as regular as possible and close to one hertz, while checking the overlap on the video feedback.

One key point, which might appear trivial to the reader but we had not considered prior to our first flight: a UAV keeps its flight elevation constant with respect to its takeoff point. In areas with strong topographic variations – as met on a glacier snout – the 100 m separating the UAV from ground at takeoff quickly shrink when the flight distance reaches 500 m over the glacier. Observing the video feedback with quickly moving features on the ground while the horizontal speed is kept constant is a signature of a shrinking elevation between ground and flight altitude, hinting at the need to reconsider the flight path before colliding with the mountain slope or the glacier surface. We have thus reached an elevation of less than 20 m over the ice surface (following the analysis of the pointcloud generated by Micmac) before wondering about the cause of this excessively fast motion of the ground on the video feedback !

Figure 1: Left: experimental conditions. The DJI Phantom3 meets the requirement of ease of transportation, fast deployment, battery lifetime and wide angle camera coverage. It has appeared well suited for this field measurement campaign. Right: canyon carved in the glacier moraine, which will be at the core of the investigation since exhibiting variable features on such short time intervals as one week. The glacier snout is visible on the bottom left part of this picture.

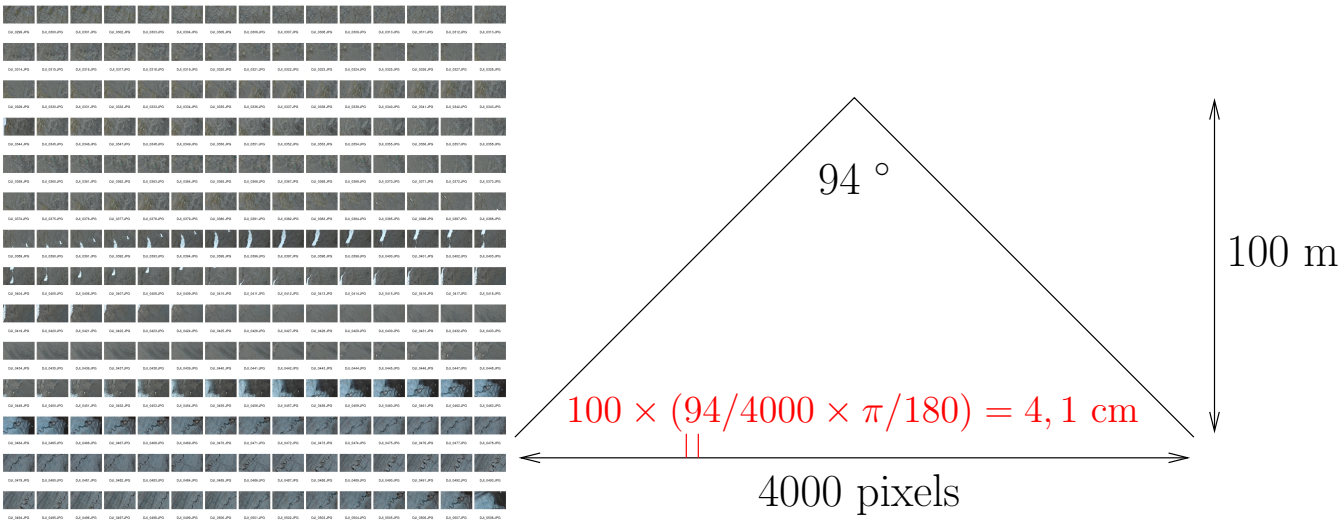


Figure 2: Left: contact sheet of a set of pictures acquired by UAV in azimuthal view, with the aim of processing for generating a high resolution DEM and an ortho-image of the area under investigation. Right: estimate of the pixel size on the ground, considering the angular aperture, number of pixels along the width of the picture, and flight height.

2 Georeferencing camera position acquired during data acquisition

Having collected pictures, we wish to process the dataset to extract a unique orthophoto – compensated for deformations introduced by ground topography – and the associated product which is the DEM. Each picture acquired by the DJI Phantom3 is tagged with its GPS latitude and longitude in degree, angular minute and second, as well as an elevation relative to takeoff altitude. These information will be used twice during the processing sequence: for selecting adjacent images to be considered to search for matching features, and for georeferencing in an absolute framework the DEM. Extracting these data in a format usable by Micmac is hence a preliminary requirement:

1. extract the three position information from each picture
2. convert degree-minute-second format to decimal degrees
3. convert from a spherical coordinate framework (WGS84 as used by GPS) to an Euclidean planar framework locally tangent to the Earth surface (UTM33N in the Spitsbergen area)
4. remove the unnecessary mean coordinate offset which will otherwise introduce rounding errors during the floating point calculation.

These operations use three tools in addition to the unix `bash` shell: `exiftool` for collecting coordinate

informations from the picture headers, GNU/Octave [1] will be used to perform computations on the coordinates, and QGis [2] for projection framework conversion.

```
for i in *.JPG; do echo  $i ; done > noms
for i in *.JPG; do echo -n $i ; exiftool $i | grep atitu | grep -v R; done > latitude
for i in *.JPG; do echo -n $i ; exiftool $i | grep ngitu | grep -v R; done > longitude
for i in *.JPG; do echo -n $i ; exiftool $i | grep Altit | grep -v R; done > altitude
cat latitude  | cut -c 56-100 | cut -d\  -f1,3,4 | sed "s/'//g" | sed 's/"//g' > latitude.txt
cat longitude | cut -c 56-100 | cut -d\  -f1,3,4 | sed "s/'//g" | sed 's/"//g' > longitude.txt
cat altitude | cut -c 56- | cut -d\  -f1 > altitude.txt
```

The three files `latitude`, `longitude` and `altitude` are filled with the position information extracted from each picture named `noms`. Argument `56` might require some tuning depending on the directory structure deployed by each user and the name of the files being handled.

The three files are used in the following GNU/Octave commands for generating the decimal degree coordinate datasets:

```
load latitude.txt
l=latitude(:,1)+latitude(:,2)/60+latitude(:,3)/3600;
save -text latitude.dec l
load longitude.txt
l=longitude(:,1)+longitude(:,2)/60+longitude(:,3)/3600;
save -text longitude.dec l
```

The two files with `.dec` extension are edited with one's favorite text editor – most probably `vi` – to remove the header, so that finally all these information are collected in one single file using
`paste  latitude.dec longitude.dec altitude.txt noms > for_qgis.txt`

Converting the coordinate dataset in decimal degrees to a flat framework is performed with QGis. Indeed, the conversion formulas are not trivial when converting from the spherical WGS84 to the tangential UTM (Universal Transverse Mercator) accounting for the non-spherical characteristics of Earth, and relying on QGis will prevent introducing errors. Using QGis, a text file with 4-columns (longitude, latitude, altitude, filename) is loaded with the icon shaped like a coma on the left of the graphical used interface. We can ease the identification of column attributions by adding in the text file a header with labels "Y X Z N" meaning latitude, longitude, altitude and name (X and Y identifiers are QGis keywords, the others will just be reproduced in the output file). Once the coordinates have been loaded, click on the right mouse button over the entry in the list of loaded datasets, `Save As` and selected the projected framework suitable for the region of interest (in our case northern Norway, hence CRS WGS84-UTM33N (EPSG:32633) – this value must be tuned for each new geographic area, for example UTM31N for most of mainland France). Finally, the file is saved with a comma separated file (CSV, named `p.csv`) and only the relevant columns are kept with
`cat p.csv | cut -d, -f1-2,5,6 | sed 's/,/ /g' | sed 's/^43//g' | sed 's/ 875/ /g' > p.fin`

In this command line, fields 1, 2, 5, 6 are the projected longitude and projected latitude, altitude and filename. These information will be used by Micmac after we add, still with one's favored text editor, a header defining the file format: `#F=X Y Z N` meaning that the # character at the beginning of a line is interpreted as a comment, followed by the order of the data (`N` is interpreted as the filename field).

Following all these preliminary steps, we have one file associating coordinates in a projected framework with each picture. This information will be a core element in the following processing sequence.

3 Assembling individual pictures

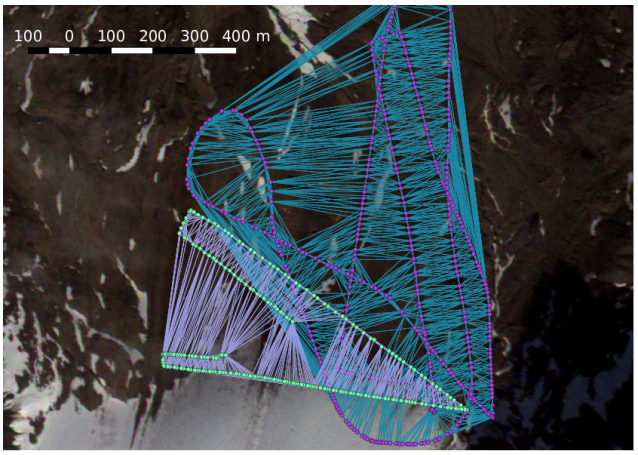
A 20-minute long flight generates about $N = 1000$ pictures. A basic pairing approach in which all possible combinations for finding common tie points would require $C_2^N = \frac{N!}{2!\cdot(N-2)!} = N \cdot (N-1)/2$ trials, a number quickly becoming huge, with for example 499500 pairs if $N = 1000$. Micmac provides the possibility of using the position at which each picture was acquired to assess whether a pair of images might contain common tie points. This option allows to restrict, in the case of our 614 processed images, analyzing only 10413 pair of images instead of 188191, a significant gain of processing time.

[1] opensource version of Matlab with excellent syntaxic compatibility for the operations we are interested in here

[2] Quantum GIS is an opensource Geographical Information System. Beyond its ability to convert between various coordinate frameworks, multiple vectorial and raster dataset processing capabilities are available, as well as for the statistical analysis of georeferenced datasets. It will be used to convert GPS coordinates to a framework locally tangent to the area being investigated, and then to insert georeferenced orthophotos and DEM over a background reference map of assessing the accuracy of the resulting computations.

Validating the sets of images selected by Micmac

We have no reason to trust the picture pairing algorithm selected by Micmac. In order to assess how valid the selection is, we wish to draw a line between each pair of image selected. We have on the one hand a file including the position of each picture, and on the other hand a file describing all the pairs. How can we generate a file telling QGis to link these two sets with lines ? One solution uses the WKT [5] language, understood by QGis, which allows describing operations between georeferenced datasets. In our case, the command for drawing a line is `LINESTRING(X1 Y1,X2 Y2)`. Hence, starting with the file `FileImagesNeighbour.xml` describing the picture pairs, we will replace the name of each picture with its coordinates:
`cat FileImagesNeighbour.xml | sed 's/<Cple>//g' | sed 's/\/Cple>//g' | grep -v S | grep -v x | sed 's/ //g' > imgpairs` which generates the file which will be given as argument to an `awk` script (named `s.awk`) aimed at compensating for the bias we have included by removing the most significant digits of the coordinates:
`{ print "sed 's/"$4"/43"$1" "875$2"/g' | \\" }` with `cat p.fin | awk -f s.awk > p.sh`
With the script `p.sh`, we replace the first line with `cat imgpairs | \` and remove the last character "\" in order to obtain a script able to generate the `wkt` description file:
`sh ./p.sh | sed 's/^/LINESTRING(/g' | sed 's/$/)/g' | sed 's/ /,/2' > couples.wkt` which is loaded by QGis thanks to the coma-shaped icon, providing the following display:



The pairs of images selected by Micmac seem consistent, we are convinced that the analysis is valid.

This signal processing step significantly reduces the number of comparisons and quickly provides a set of tie points between image pairs:

```
mm3d OriConvert OriTxtInFile p.fin jmfgps MTD1=1 NameCple=FileImagesNeighbour.xml CalcV=1 \
     ImC=DJI_0115.JPG NbImC=25
```

tells Micmac to create the orientation directory `Ori-jmfgps` (which will be used, in the end, to convert thanks to scaling and translation, the point cloud generated by Micmac in an arbitrary framework, to an absolute geographic framework) as well as a file named `FileImagesNeighbour.xml` describing picture pairs, by using (`OriTxtInFile`) a configuration file `p.fin`.

On the other hand, a reference picture is selected – here DJI_0115.JPG – whose neighbors will be used for a first estimate of the optical properties of the lens and the camera. Selecting this reference picture should be done while complying as much as possible with the requirement of taking a picture of a corner when assessing the lens properties – as much as a corner might be visible in the field under investigation. A valley between hills seems appropriate. The `PATC` variable generated at the end of this processing step includes the list of picture names appropriate for this calibration step. We add to the definition of the `PATC` variable definition another variable named `P` which includes all the pictures stored in the current directory, assuming we have already removed all the unnecessary images (blurred, oblique views) and only kept those worth processing, in azimuthal view only:
`P=".*JPG"`

Searching for tie points is finally performed with
`mm3d Tapioca File "FileImagesNeighbour.xml" -1` followed by the calibration of the lens and camera properties (modify the lens model – `RadialStd` – with a FishEye if a GoPro camera or an older DJI UAV with a wide angle lens is used):
`mm3d Tapas RadialStd "$PATC" Out=Cal` This last step is critical since its success is very dependent on the selection of the reference picture: several trials are sometimes needed for the model to converge (residual below < 1 pixel) and prevent divergence (residual equal to NaN).

Finally, this calibration is applied to all pictures in order to assess the position and orientation of the camera as each picture was taken

`mm3d Tapas AutoCal "$P" InCal=Cal Out=Init`

We now use the orientation directory to convert the arbitrary framework in which Micmac completed its calculations, to an absolute geographic framework `CenterBascule "$P" Init jmfgps tmp` before generating a coarse point cloud for validating the consistency of the location at which pictures were acquired and the point cloud shape (Fig. 3) `mm3d AperiCloud "$P" tmp`

One might consider, after reaching the geographic framework following the use of `CenterBascule`, to create a new orientation weighting position information extracted from the GPS (with its associated uncertainty) and photogrammetry thanks to `Campari`:
`Campari "$P" tmp tmp-Campari EmGPS=[jmfgps,2] FocFree=1 PPFree=1` In this case, the coarse point cloud and the following operations would be performed on the orientation directory `tmp-Campari` instead of `tmp`.

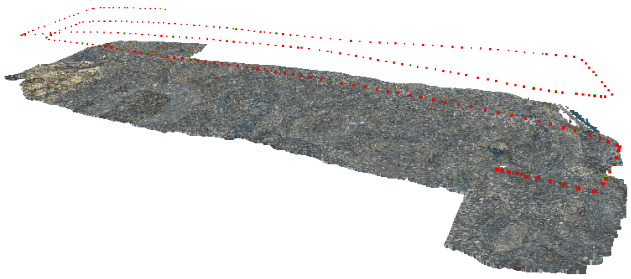


Figure 3: Coarse point cloud, including the position of the camera, as found following photogrammetric image processing, as each picture was acquired. The red dots are the camera positions, the grey coarse point cloud is the surface resulting from processing the pictures.

4 Products obtained from image processing

4.1 Orthorectified image

Having validated the consistency of the coarse point cloud (`meshlab` or `cloudcompare`), we compute the dense point cloud with
`mm3d Malt Ortho "$P" tmp "DirMEC=Resultats" UseTA=1 ZoomF=2 ZoomI=32 Purge=true AffineLast=false`
before concluding with the orthorectified pictures (compensated for topographic effects by assembling adjacent projected pictures) `mm3d Tawny Ortho-Resultats/ RadiomEgal=0` Finally, the dense point cloud is colored with the orthorectified picture:
`Nuage2Ply Resultats/NuageImProf_STD-MALT_Etape_7.xml Attr="Ortho-Resultats/Ortho-Eg-Test-Redr.tif"`

The resulting images are huge – so large that their size is beyond the one handled by the TIF format – and require several giga-bytes for storage in the case of the several hundreds of pictures we have processed. Obviously, with a ground based pixel size of 4 cm, and an area under investigation 800 m long, the image width will be 20000 pixel ! Pictures 20000×30000 pixel large occupying 1.2 GB are often achieved since no compression is included in the TIF format used by Micmac. We convert the TIF format to PNG – with its more efficient storage despite a lossy compression algorithm – and assemble the resulting files in one single orthoimage since PNG does not seem to be limited in the represented picture size. ImageMagick is most efficient in performing such operations, despite some issues if the `/tmp` directory is too small to handle the huge datasets (for example if it is a modest ramfs filesystem). We export a variable telling ImageMagick to use our home directory for storing temporary files `export MAGICK_TMPDIR=$HOME` before running the conversion and assembling pictures:
`convert Ortho-Eg-Test-Redr_Tile_0_0.tif Ortho-Eg-Test-Redr_Tile_0_0.png` in the `Ortho-Resultats` directory before assembling pictures with

```
montage -tile 1x2 -geometry +0+0 Ortho-Eg-Test-Redr_Tile_0_0.png \
   Ortho-Eg-Test-Redr_Tile_0_1.png out.png
```

The resulting pictures remain huge in terms of pixel size, but at least can be handled by most viewers (`display`, `geeqie`) since their storage size was reduced.

Each obtained picture is associated with a positioning file indicating the coordinates of the top-left corner (also known as the north-western corner – the last two lines of the file) and the pixel size: the file with extension `tfw` is renamed `pngw` for the picture we have just assembled. The offset we have removed when generating the initial position file must however be restored: we had removed the prefix 43 to longitudes and 875 to latitudes, which we take care to insert back here. In our case, a typical position file looks like

```
0.045
0
```

```
0
-0.045
438839.62
8760127.52
```

Notice on lines 1 and 4 the pixel size (4.5 cm, as expected from our rough geometric calculation), and that the latitude being of the order of 10^5 m, adding 8750000 results in the first digits becoming 876 in the last line.

A rough analysis seems at first glance to hint at the proper overlap of orthorectified images when comparing with a reference satellite image (Fig. 4). However, free satellite picture sources (e.g. Landsat) provide too coarse a pixel size to assess the resolution we are interested in: the typical 30 m-large pixel only allows for a coarse assessment of the accuracy of the processing chain we have described so far, well below the target sub-meter accuracy.

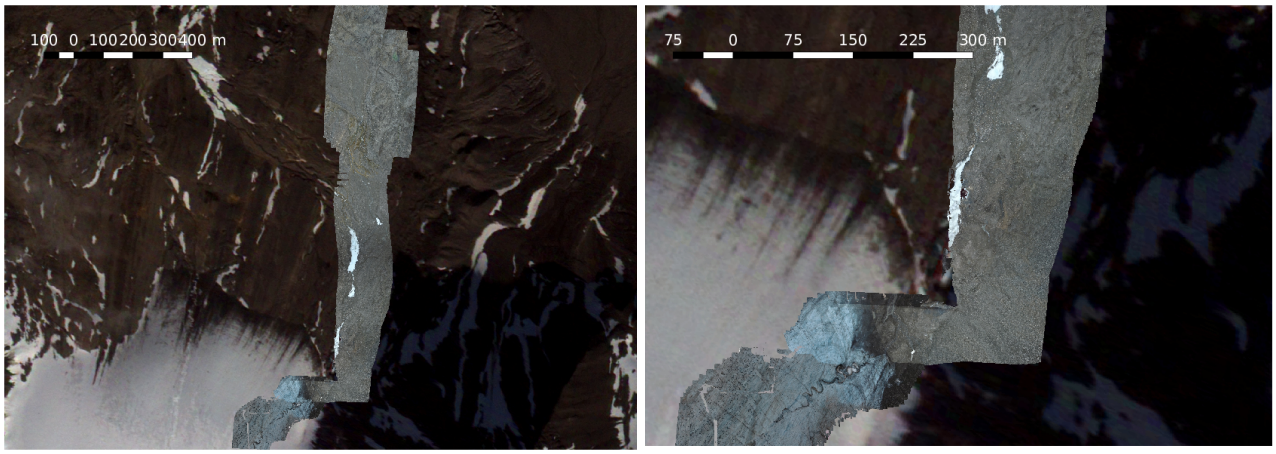


Figure 4: Orthorectified picture overlapping a background satellite image. Notice that the black background of the orthorectified picture was eliminated by selecting `Transparency` → `Add` and the color `0 0 0` to be considered as transparent in the QGis attribute editor for this picture. Notice the resolution improvement of the UAV-based orthoimage with respect to the satellite image considered as reference.

We will now aim at quantitatively estimating the reproducibility of the measurements, by comparing datasets acquired several days apart.

4.2 Digital Elevation Models (DEM)

Positioning the DEM, available at `Resultats/Z_Num7_DeZoom2_STD-MALT*`, rises the same issue with a GeoTIFF file to be modified in order to add the bias introduced initially. Furthermore, a `XML` file associated to the DEM tells us what the offset and scaling factor between the TIF integer pixel value and real altitude are:

```
<OrigineAlti>9.675</OrigineAlti>
<ResolutionAlti>0.09</ResolutionAlti>
```

When loading the image in QGis (raster dataset icon), the DEM is properly located, but altitudes are erroneous.

Scaling and translation on the altitude data are performed using the `Raster` → `Raster Calculator` for creating a new layer (`Output layer`) in GeoTIFF format which includes the affine transformed input dataset (Fig. 5).

Fig. 5 exhibits on the left a satellite image of part of the moraine area under investigation with some topographic features identified from space, and on the right the DEM, reproducing accurately the hydrographic network. Since the latter result is the result of our processing steps, we can address the resolution and accuracy issues.

5 Comparing multiple measurements acquired at intervals of a few days

Although the GPS constellation resolution is in the meter range for a given satellite configuration, the long term accuracy is only in the tens of meter range and hence introduces an unacceptable offset between multiple datasets acquired over a few days. Hence, co-locating the orthophoto and DEM datasets separated by multiple days, when the constellation configuration and ionospheric conditions have significantly evolved, is mandatory. Since we have omitted deploying georeferenced Ground Control Points during picture acquisition, considering

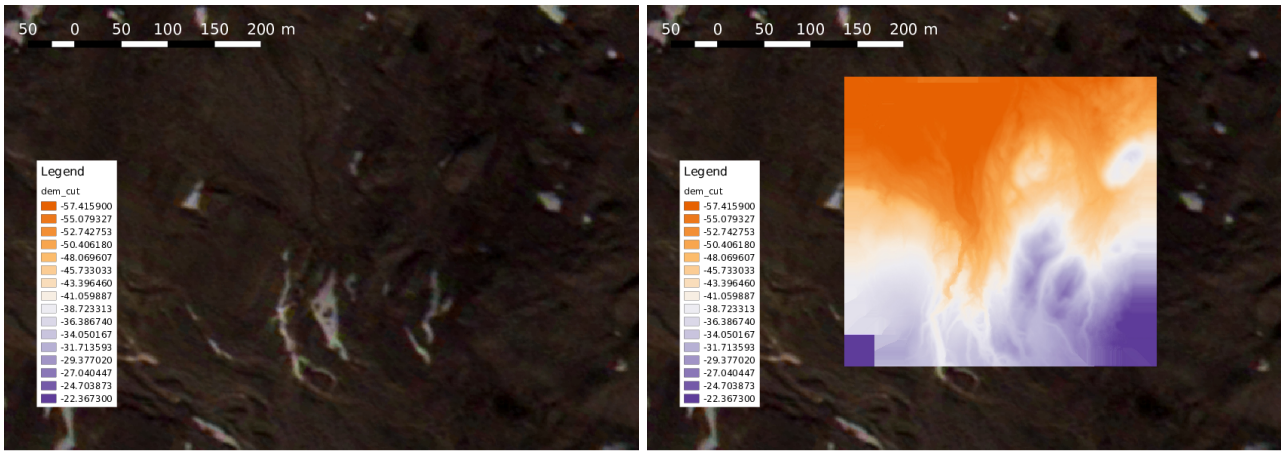


Figure 5: Left: satellite image of the moraine. Right: DEM positioned only using the GPS position of the top-left corner, without manual positioning . The consistency of the observed features is best seen on the boundary between the DEM and the background image of the right figure.

the timing constraints of the field trip, we will attempt to compensate for varying dataset positioning in a post-processing step for matching relevant features seen on the ground. Hence, even if our datasets are not properly located in an absolute geographical framework, they will at least be consistent one with respect to the other, allowing for a comparison (Fig. 6).

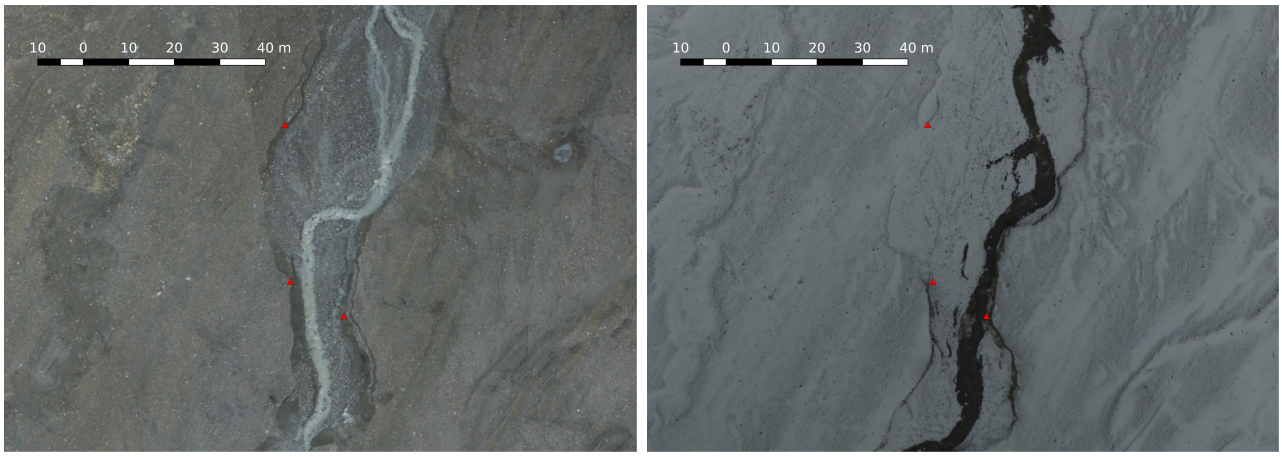


Figure 6: Two orhophotos generated from pictures acquired one week apart are co-located. Red dots are reference points (`shapefile` including such a dataset) allowing to match such relevant features assumed located at a fixed position between the two pictures. Notice the evolution of the stream path and braiding between the two measurements.

Two strategies are available for this matching step. The former uses a plugin named `rasmover` provided for QGis: despite running as a graphical interface and being trivial to use (just slide the arrow indicating the displacement vector over the image being processed), this approach prevents recovering a quantitative information on the translation vector (how far was the picture moved along the arrow). We hence favor the latter approach which directly modifies the `world` file header to add the offsets required to achieve proper matching of both datasets. We hence control the consistency of the displacement we apply on the various images and prevent inconsistent manipulations to which graphical interfaces are prone.

Following the fine positioning of the orthophotos, and hence the associated DEMs by applying the same corrections to the world files, comparing two datasets acquired one week apart is demonstrated in Fig. 7. We observe that the canyon on which we have focused our investigation does not exhibit significant depth variations since the elevation differences are close to a constant value (non-zero since we have not subtracted the UAV takeoff altitude, different during the flights performed at the two dates): light colored pixels are distributed around 70 cm error of the lower end of the elevation difference distribution. A slight offset is left between the position of the two DEMs, as seen on the slopes of the canyon which do not perfectly overlap, with the western slope in blue and the opposite eastern slope in red. However, a dark red area is visible (indicated by $dz \sim 3$), associated with a 3-m deep landslide (fourth colorbar segment when starting from the bottom), a thickness consistent with the depth of the adjacent canyon. This analysis is confirmed with a cross-section using the `Terrain Profile` plugin (Fig. 7, right).

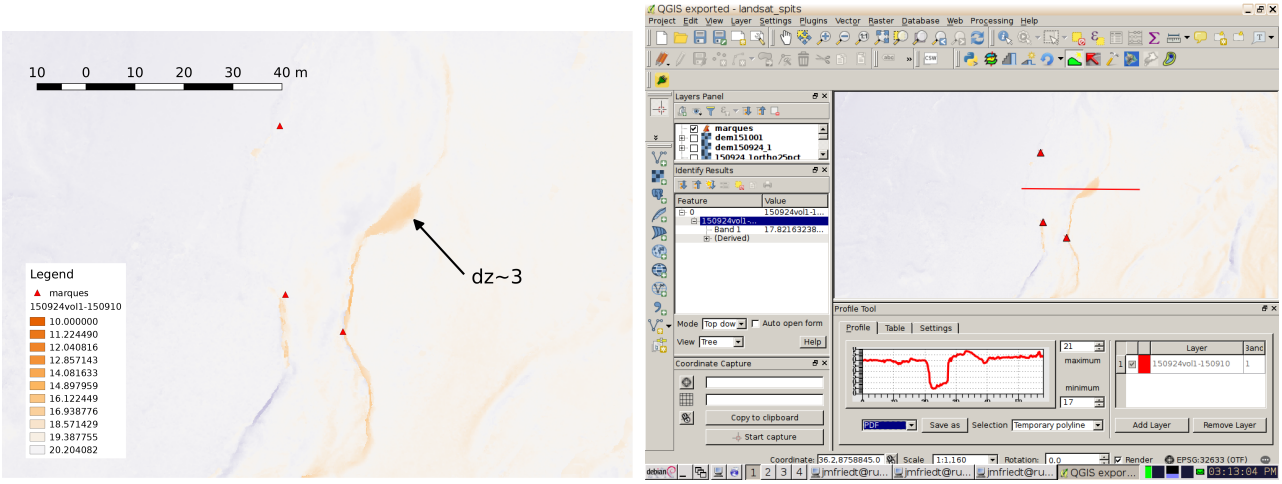


Figure 7: Analysis of the error between DEMs acquired one week apart. Left: notice the 3-m deep landslide, consistent with the depth of the neighboring canyon, on the difference of DEMs map. Right: screenshot of the QGis tool `Terrain Profile`, with the red line indicating the profile under investigation.

6 Conclusion

We have considered an acquisition and processing sequence in order to generate digital elevation models with decimeter resolution. We have compared the accuracy of the collected data and, having concluded for the need to manually match positions as no georeferenced ground control point was positioned, we have observed a landslide which occurred during the one-week interval between two flights.

This processing approach, developed initially over a three-month duration following the dataset acquisition, now requires about one day of processing on a quad-core Xeon computer clocked at 2.13 GHz (8 cores, 32 GB RAM) for each flight, assuming 700 usable pictures were acquired during each 20-minute long flight.

Some of the datasets are available at `http://jmfriedt.free.fr/dji_micmac`, and a second example closer to Wester European environments with a dataset collected in the park of the Besançon Observatory (emphasizing the issues related to trees and plants) is presented at `http://jmfriedt.free.fr/dji`.

References

[1] J.-M. Friedt, *Reconstruction de structures tridimensionnelles par photographies : le logiciel MicMac*, Open-Silicium **12** (Sept-Oct-Nov 2014) [in French]

[2] M. Pierrot Deseilligny, *MicMac, a free open source solution for photogrammetry*, RMLL 2015 (video available at `https://2015.rmll.info/micmac-une-solution-libre-de-photogrammetrie?lang=en`

[3] M. Nolan, C. Larsen, & M. Sturm, *Mapping snow depth from manned aircraft on landscape scales at centimeter resolution using structure-from-motion photogrammetry*, The Cryosphere, 9, 1445–1463 (2015), available at `www.the-cryosphere.net/9/1445/2015/`

[4] Y. Lin, J. Hyyppä, & A. Jaakkola, *Mini-UAV-borne LIDAR for fine-scale mapping*, IEEE Geoscience and Remote Sensing Letters, **8** (3), 426–430 (2011)

[5] `http://gis.stackexchange.com/questions/111519/draw-lines-between-two-points-in-qgis`