Photogrammetric 3D structure reconstruction using Micmac

J.-M Friedt, 5 septembre 2014

We consider a complex image processing problem, namely the reconstruction of the 3D structure of objects pictured from different points of view. Amongst the tools available to achieve such a result, the most serious solution for achieving a quantitative model of the object (dimensions in real units, opposed to a point cloud graduated in pixels) is a software provided by the French National Geographic Institute (IGN) designed to general digital elevation and wall models : the Apero and MicMac software libraries.

In this context, this presentation is a complement to a previous study on processing spatialized information requiring a digital elevation model (DEM) [1] : here, rather than being a user of such a dataset, we will become producer of the dataset (even though we will see that the path to reach a dataset quality comparable to those available freely on the web is still long). Without pretending to compete with professional tools such as LiDAR (the optical version of the RADAR measuring the time of flight of an electromagnetic pulse in the visible or infrared wavelengths) – a costly tool complex to handle – digital image processing provides a solution easily deployed yet with lower resolution, meeting multiple requirements including the fun objectives demonstrated here.

Generating 3D models of objects from digital pictures is an increasingly popular topic [2], probably related to the wide availability of nearly infinite processing power and of excellent quality yet low cost optical sensors [3]. The keyword describing this image processing strategy – including not only the signal processing step but also rules for acquiring the digital images and, at then end, the tools for displaying the point clouds – is *Structure from Motion* (SfM).

1 Basics of SfM

Multiple tools exist for acquiring images for 3D reconstruction of an object structure. Arguably the most popular is the Microsoft Kinect, designed to project a cloud point in the infrared wavelengths (invisible to the naked eye) for a reconstruction of the illuminated scene. This strategy is only applicable indoors and for a reduced range of a few meters, and is hence not applicable for our purpose of generating DEM on a range of a few tens of meters to kilometers. Professional mapping tools using the time of flight of laser pulses (LiDAR) are financially inaccessible to the amateur. We will thus consider using the many optical sensors available – smartphone, webcam, digital camera – for generating 3D point clouds representing the scene observed from different points of view.

As is always the case when considering image processing, the core issue of the reconstruction of the 3D structure of an object imaged from multiple points of view lies in the quality of the initial images. Not complying with the requirements of image acquisition induces great chances of the failure of the subsequent processing (this topic will be discussed layer, section 2). Having acquired images following principles exactly opposite to those used for obtaining panoramas – rather than taking many images in different directions from a given site, a single object is aimed at from different points of views – the aim is to identify common points visible on all the images. We had already discussed the concept of cross-correlation [4] for recovering identical patterns on different images and compute the translation of the object between the two frames, but such algorithms are extremely sensitive to rotation and scaling of the image sub-frames. An efficient combination of various processing steps yielding insensitivity to scaling (multi-scale analysis) and rotation has been devised when producing the SIFT algorithm [1] [5] aimed at filling these gaps of the cross-correlation. Identifying relevant reference features visible on all images is a core aspect of the signal processing chain, but is only the first step of the point cloud reconstructing which also requires the identification of the optical properties of the lens and the position of the camera when the shots were taken, compute the position of the each point visible on the pictures thanks the geometrical properties of the scene and the camera, and finally view the resulting point cloud (typically made of a few hundred thousand of points) [2] [6].

1. `http://en.wikipedia.org/wiki/Scale-invariant_feature_transform` or, more detailed, `http://omiv2.u-strasbg.fr/imagemining/documents/IMAGEMINING-Stumpf.pdf`
2. without the oral presentation, the slides at `http://fad.ensg.eu/moodle/course/view.php?id=315` are challenging to understand but at least provide some hint at the general workflow

Multiple implementations of these concepts exist, depending on the context and the objectives : we have identified the Python Photogrammetry Toolbox (PPT), bundler [3] as well as the solution we have to select the investigate further, MicMac (Multi-Images Correspondances, Méthodes Automatiques de Corrélation [4]) from the French National Geographic Institute (IGN) (`logiciels.ign.fr/?-Micmac,3-`). Beyond the serious development environment of the geography institute and the point cloud quantitative analysis framework, and author starting his documentation by explaining that the software was written because he "thought it would be fun to be able to make 3D models from my holidays pictures" [7] can only have produced an excellent application meeting our requirements. We will see in these pages that we shall not be disappointed. The documentary "Les derniers secrets de l'armée de terre cuite" [5] hints at the fact that this technology revolutionizes the ability to store 3D archives : without reaching such an enthusiasm, it indeed appears as a technology accessible to the motivated amateur not requiring and money investment other than the time it takes to learn how to use the available tools as described here. Access to excellent quality optical sensors for a minute cost, nearly infinite processing power in each personal computer, and the wider availability of flying equipment such as drones, combine the requirements for the field of 3D point cloud reconstruction from pictures taken from different locations to become available in a wide range of fields [8, 9, 10] beyond the restricted community of computer vision.

The main inspiration of this tutorial is the brief documentation available at `http://combiencaporte.blogspot.fr/2013/10/micmac-tutoriel-de-photogrammetrie-sous.html` [in French], with the addition of the MicMac documentation provided with the online archive. Installing the opensource software is trivial by fetching the archive using `mercurial` [6] and following the installation method described in the `README` file, which no one ever reads so is summarized here : starting from the source directory `$MICMAC`
`mkdir build && cd build && cmake ../ -DWITH_QT5=1 && make && make install`
will install in `$MICMAC/bin` the executable binaries (to be added to the `$PATH`) and in `$MICMAC/lib` the associated libraries, so that this latter path should be added at the beginning of the `LD_LIBRARY_PATH` variable. Reading the documentation, available in `$MICMAC/Documentation/DocMicMac.tex`, is mandatory throughout this presentation. Furthermore, short hints at the usage of each command are available : `mm3d -help` provides a list of all MicMac tools, while `mm3d tool -help` provides a detailed list of all options accepted by a tool.

2 Getting started : picture acquisition methods

We will see below that MicMac is able to recover all picture acquisition conditions (positions of the camera, lens properties, lens focal length) even when providing random initial parameter guesses, but reaching such an ambitions goal requires meeting some requirement : all pictures must be taken with the same camera [7], under the same lighting conditions (avoid using a flash which induces variable cast shadows depending on the point of view), and most important with *same focal length* (or magnification). The quality of the optical sensor does not matter : we illustrate this text with examples acquired from a mobile phone single lens camera, a webcam, or a compact digital camera (all of which exhibit picture quality far from those found with high grade reflex cameras), but complying with the 80% overlap requirement from one image to another, and constant focal length. Detailed and rigorous descriptions of the picture acquisition conditions are described in `http://www.tapenade.gamsau.archi.fr/TAPEnADe/PR_facades.html`, but we will see later that when using scenes from aerial views found in television documentaries, strict compliance with these rules is not always needed. However, we *strongly encourage* the reader to become familiar with the concepts presented in this manual which, despite an austere title, provides rules for acquiring pictures (sometime hard to meet) in order to optimize chances of the computation to converge : most significantly, page 9 (figure 11) shows how to best take pictures of a building by taking pictures from different angles, with some used to stitch the various views used to

3. `www.cs.cornell.edu/~snavely/bundler/`
4. `http://www.tapenade.gamsau.archi.fr/TAPEnADe/Tools.html`
5. `http://www.arte.tv/guide/fr/050492-000/les-derniers-secrets-de-l-armee-de-terre-cuite`
6. `hg clone https://culture3d:culture3d@geoportail.forge.ign.fr/hg/culture3d` : all the demonstrations in this document have been achieved with version 3642 dated juin 20th, 2014
7. Processing multiple picture series of the same scene taken with different cameras or lenses is possible – for example zooming on near field details seen on wide angle far field general views – but each picture set is processed individually during the first steps, especially for identifying the geometrical properties of the lenses.

generate the point cloud...

Before starting the actual processing steps, we must fill some dimensions concerning the optical sensors of the instruments used to acquire images if such information is not yet available in the MicMac database. The fields are added in the `$MICMAC/include/XML_User/DicoCamera.xml` file within the source tree by adding, for example for the Panasonic TZ-10 and the Samsung Galaxy S3 mobile phones, some of the fields found in the EXIF header (identifier of the camera) and the dimensions (in millimeters) of the optical sensor :

```
<?xml version="1.0" ?>

<MMCameraDataBase>
<!-- Pana TZ10 DSLR -->
   <CameraEntry>
        <Name>  DMC-TZ10  </Name>
        <SzCaptMm>  6.0 4.0 </SzCaptMm>
        <ShortName> DMC-TZ10 </ShortName>
   </CameraEntry>

<!-- Telephone Samsung S3 -->
   <CameraEntry>
        <Name> GT-I9300 </Name>
        <SzCaptMm>   4.54 3.42 </SzCaptMm>
        <ShortName> GT-I9300 </ShortName>
   </CameraEntry>
</MMCameraDataBase>
```

The information we provide in the `SzCaptMm` field is the dimension of the optical sensor in mm unit (as often found on the web), while the `GT-I9300` field provides the name of the digital camera as found in the EXIF header (`exiftool photo.jpg | grep Model`).

We will now described a detailed sequence of the four processing steps needed to run the MicMac tools : **identify relevant features** on all picture pairs, compute the **position and optical properties** of the instrument used to take the pictures thanks to these common features, visualize the result of this computation in order to **validate** the quality of the result of these processing steps (a camera located in an aberrant position will yield failure of the next processing steps and hint at erroneous results from the previous computation steps or incorrect picture acquisition conditions), and finally generate a **dense point cloud** by positioning in space the pictures visible on the pictures displaying a given object.

3 First example ... stand in the corner

Let us start with the result we wish to achieve : a point cloud of a corner featuring good contrast thanks to wood fibers and a map attached to the wall (Fig. 1), ideal conditions for the reconstruction with features easily identified on multiple pictures and an appropriate depth [8]. We will see that reaching this result is achieved by running sequentially four commands : `Tapioca`, `Tapas`, `Malt` and `Nuage2Ply`. Most applications use at best the multi-threading capability of modern processors, thus significantly reducing the computation duration by calling at best each processing step.

The most important aspect of using MicMac is understanding the philosophy behind the software : starting from digital or digitized pictures in the arbitrary plane of the optical sensor, the aim is to run the dataset through various framework orientations in order to convert the 2D information stored in each picture ti a 3D point cloud representation. Each step is associated with the creation of an orientation related directory, prefixed with the `Ori-` naming convention. Hence, each processing step will use as input the name of the directory including the results of the previous processing step, and as output the name of the directory containing the result of the new processing step. When the directory name is not provided, the default behavior induces a naming convention not always obvious to understand in the beginning. Hence, starting by calibrating the optical properties of the lens on a dedicated subset of the available pictures, we indicate that the result must be stored in a `Ori-Calib` directory by providing as an argument to the appropriate command `Out=Calib`. During the next processing step aimed at

8. All the contact sheets have been generated with ImageMagick by using `montage -geometry 120x120+1+1 -tile 10x1 -label %f *.JPG contact.jpg`

FIGURE 1 – Top : four pictures of a corner reconstructed (bottom) as a 8.9 million point cloud. The bottom-right map includes the camera position when each image was taken, visible as a red rectangle with increasing focal length illustrated as a sharper green triangle. The summit of each triangle is located at the focal point of the camera. The white-colored area behind the triangle next to the top left corner represents a region which is not visible on the reference picture (P1030124.JPG) and in which no depth information was associated on the point cloud.

applying this calibration step to all the acquired pictures, the previous result is exploited by providing as arguments `InOri=Calib Out=Next` in order to store the result in `Ori-Next`.

The initial processing sequence is directly inspired from `http://combiencaporte.blogspot.fr/2013/10/micmac-tutoriel-de-photogrammetrie-sous.html`. Summarizing this excellent tutorial :

1. image processing starts by identifying the relevant features on picture pairs (so called homologous points) using the `Tapioca` tool. The MicMac manual [7, section 3.3.1] tells us that the result of this processing step is located in the `Homol` directory, whose files are easily analyzed since they contain the coordinates of the starting and ending displacement of all the relevant features identified on each picture pair (ASCII formatted files if the `Tapioca` command was appended with the `ExpTxt=1` option). As an example of such a result, we provide in Fig. 2 the set of displacement vectors between homologous points identified by the SIFT algorithm on images acquired while flying over Spitsbergen (only one in every ten point identified as relevant was displayed in order for the figure not to be too crowded).

   This example hints at a poor resolution when processing surfaces homogeneously covered with snow or ice missing relevant features.

   We actually run the following commands :

   ```
   mm3d Tapioca MulScale ".*jpg" 300 1000 ExpTxt=1
   ```

   Although POSIX regular expressions (which do *not* comply with the same rules than the shell regexp!) can be used for defining the file subset to be processed, we have used the convention of calling raw pictures with the `.jpg` extension, and temporary files (mask, scaled raw reference pictures) with the `.JPG` extension. Using this command, common relevant features are searched on all picture pairs at various scales provided as argument, in this example with the longest axis ranging from 300 to 1000 pixels in order to reduce computation time : replacing 1000 with -1 would induce a search all the way to the maximum resolution of the picture, with a lengthening of the computation time. It seems commonly accepted for the upper resolution range boundary to be about one third of the original picture width,

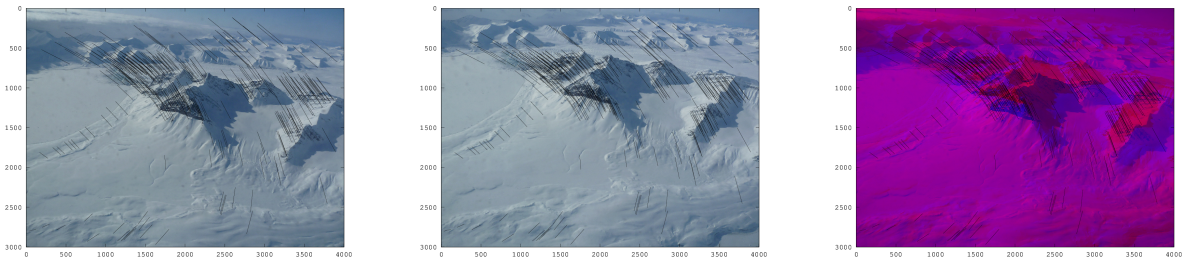


FIGURE 2 – Homologous points identified on sequential pictures (left and middle), and overlap of the two color-encoded pictures demonstrating that relevant features are indeed identified on both ends of the motion vectors, despite the scaling and rotation between the two datasets. Homologous point coordinates are provided in the `Homol` directory created by `Tapioca`, in ASCII format accessible by GNU/Octave if the `ExpTxt=1` was provided.

2. common relevant points having been identified, the optical properties of the lens and the optical sensor are computed by `Tapas`. In a first step, a subset of all pictures is processed to characterize the geometric properties of the optics : the resulting model is then applied to the whole dataset :

   ```
   mm3d Tapas RadialStd ".*jpg" Out=Cal1
   mm3d Tapas AutoCal ".*jpg" InCal=Cal1 Out=Ori1
   ```

   `Cal1` and `Ori1` extensions yield the creation of the `Ori-Cal1` and `Ori-Ori1`i directories : these directories include the computation results needed to feed the next processing step,

3. the computed position and orientation of the camera when each picture was taken are visualized on a rough, low-resolution cloud point including the camera position. This cloud point including camera position markers is here called `PosCams.ply` :

   ```
   mm3d AperiCloud ".*jpg" Ori1 Out=PosCams.ply
   ```

4. having validated that the camera position is correct with respect to the observed structure, the "true" computation resulting in a dense cloud point is launched by

   ```
   mm3d Malt GeomImage ".*jpg" Ori1 "Master=P1030158.jpg" "DirMEC=Resultats" UseTA=1
   ZoomF=4 ZoomI=32 Purge=true
   ```

   which requires as an argument a reference image to define which pixels must be located in space. This image provides the point set (in two dimensions) for which the third dimension will be computed, resulting in the point cloud. In order for the computation not to last too long by looking for all the depth positions of the full scale picture, this computation is performed on scaled versions of the master image with ratios of 1/32 to 1/4 of the original dimensions, and putting the results in the directory "Resultats",

5. the result of the computation is converted to a point cloud by running

   ```
   mm3d Nuage2Ply "./Resultats/NuageImProf_STD-MALT_Etape_6.xml" Attr="P1030158Zoom4.JPG"
   ```

   where the scaled version of the original image by a 1/4 factor is obtained using the command
   `convert image.jpg -scale 25\% imageZoom4.JPG`
   If the result is obtained by using a master image with a different scaling factor, both the `ZoomF` argument (1 for the full resolution, which then takes as `Attr` argument of `Nuage2Ply` the same `Master` image than the one used by `Malt`) and the step index of `NuageImProf_STD-MALT_Etape_6.xml` (8 for a full scale image) must be adapted to be consistent. An alternative approach, instead of creating a new file with the scaled master image, the `Attr` argument can be the original image and the scaling factor is then provided independently as `RatioAttrCarte=4`.

Using this basic processing chain allows for validating each step and, after becoming more familiar with the software, identify some of the corrective steps if necessary. As an example, `Tapas` provides multiple models of optical properties of lenses with increasing numbers of parameters. The most complex

model – `RadialExtended` – is the most flexible but will not always converge, while using a simpler model – `RadialBasic` – maximizes chances of converging by reducing the number of degrees of freedom. `RadialStd` seems to be the model to be used in most conditions. This lens parameter identification step appears as a critical achievement which might yield to the failure of the whole processing chain if the pictures do not comply with some requirement (constant focal length, overlap of more than 80% between pictures, sharp images).

Understanding some of the output fields of `Tapas` is useful : beyond the cryptical error messages when a parameter is missing (most often related to missing information in the EXIF header of the pictures being processed), the only meaningful message we understand in the processing chain is the one provided by `Tapas`. Values provided as

```
RES:[P1040263.JPG] ER2 0.559285 Nn 100 Of 179 Mul 115 Mul-NN 115 Time 0.00602202
RES:[P1040273.JPG] ER2 1.1225 Nn 96.5096 Of 573 Mul 311 Mul-NN 311 Time 0.0184491
```

include the residual error of the point cloud (fourth column) while model parameters are being tuned (this field should ideally become null, and reaching a value around one pixel is a reasonable target, hence a value around 1), the ratio of the number of points being analyzed (6th column) within the homologous point dataset between two images (here 100% and 96,5% respectively), and the number of homologous points being considered (8th column). Periodically, when the residual error becomes small enough, one of the parameters of the optical properties of the lens is set :

```
LIB  DR1
```

and the algorithm steps to the next parameter. This analysis of each processing step – check the camera position, check that the lens model has converged by reaching an error below one pixel, check that the correlation maps `Correl*` are bright – hint at *not* automating the processing steps by calling the various tools in a `Makefile` script, but rather call each processing step manually and visually validate the result before running the next step.

4 Visualization tools and point cloud manipulation

Two opensource tools seems most appropriate for handling the huge datasets generated by the processing steps described above : `meshlab`[9] and `CloudCompare`[10]. The former is available as a binary package for Debian GNU/Linux, the latter is compiled easily and deserves further investigation due to its ability to match two point clouds, estimate the difference between two datasets, and manipulate these point clouds to only keep a subset of the points. In order to demonstrate the last topic, we consider pictures including clouds which MicMac positions at a very large distance from the targeted scene (Fig. 3). Initially, the result seems unusable, but searching closely near the convergence point of the vanishing lines indicates that the model of the area near the East train station in Paris is indeed available, but shadowed by a huge number of unusable points (clouds). Two approaches will correct this issue : define a processing mask, or remove the unusable points. The latter approach is considered first and we will conclude with the former.

> Meshlab is a flexible tool ... once we have become familiar with its use. Amongst the shortcuts making life easier, double clicking on a point defines this spot as the new center for rotating and scaling the view, while Alt+wheel changes the size of the pixels of the point cloud being displayed.

CloudCompare is a powerful tool for manipulating point clouds, including the removal of unwanted subsets of the dataset : in this case, the clouds in the sky which were considered by MicMac very far from the scene we are interested in and which has hence become unusable. The manipulation tool only become active once the point cloud name has been selected in the left menu (name highlighted in blue) : is is reached by selecting `Edit` → `Segment` which allows selecting some points in the point cloud to be kept or removed. Using this tool successively from multiple points of view, the resulting reduced dataset is confined to the scene of interest and easily visualized in `meshlab`.

Alternatively, especially to save processing time, the dense point cloud correlation `Malt` step can be restricted to the interesting areas. This software can indeed use a mask defining the regions of interest

9. `meshlab.sourceforge.net`
10. `www.danielgm.net/cc/`

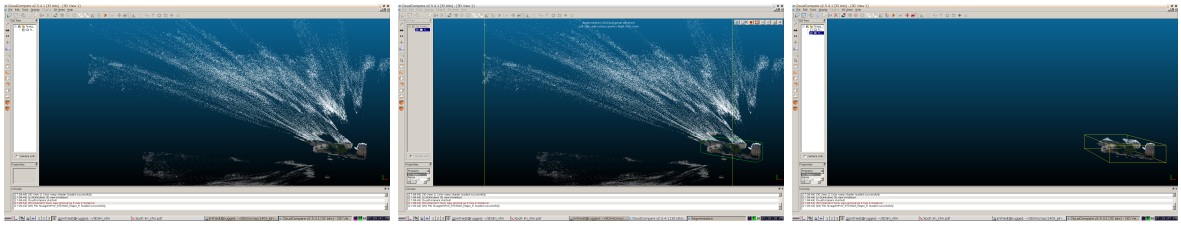


FIGURE 3 – Left : point cloud representing the area close to the East train station in Paris, including in the foreground points identified as near to the balcony from which pictures were taken, and in the sky the clouds located far from the scene (one of the original images used for this processing is displayed on the left of Fig. 4). Middle : the point cloud is selected (highlighted field in the left menu) and the tool for cropping the dataset allowed for selecting only the area of interest. Right : having removed the useless points, the new cloud point subset can be viewed in `meshlab` with a center of rotation and scaling located in the region of interest.

(shown in white) and exclude the areas of the reference image that the user knows not to include useful information. This mask is drawn either using the Gimp as described in `http://combiencaporte.blogspot.fr/2013/10/micmac-tutoriel-de-photogrammetrie-sous.html` (saving the mask in an uncompressed TIF format, using a binary black and white scale by selecting `Image` → `Mode` → `Indexed`) or the dedicated tool provided by MicMac under the name `SaisieMasq` (left button to define the polygon in which the area to be kept will be drawn in green, shift-left button to close this polygon, and finally ctrl-right button and Exit to save the mask `.tif` file and the associated `.xml` description). Although unsophisticated, the latter tool generates the needed XML file which must otherwise be filled manually. At the end, the mask defines the areas to be processed in white and shortens the processing duration by only handling the regions of interest (Fig. 4). The default behavior of `Malt` is to use the mask information which can be explicitly selected by setting the `UseTA=1` option. The default extension of the file is `_Masq`, which can be modified with the `MasqIm` option of `Malt`.

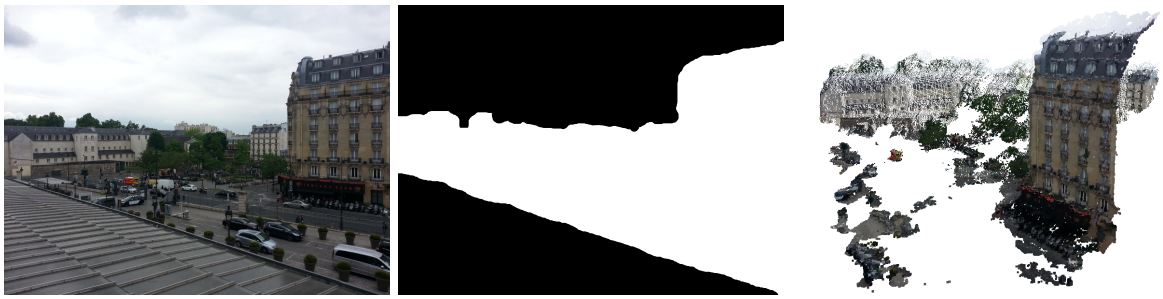


FIGURE 4 – Left : one of the original image used for generating a model of the area surrounding the East train station in Paris. Notice on the foreground the balcony and clouds in the sky which yield a hardly usable point cloud result. Middle : mask removing (black regions) these useless areas. Right : the resulting point cloud when the mask information is used, reducing the point cloud to the region of interest.

5 Practical cases

We highlight here some practical cases which motivated this study. Our aim has never been to reach perfect results, but rather use all available image sources, even of poor quality, to assess how robust the tools provided by MicMac are. We can already state that the result is impressive.

The reader is advised, both for learning and becoming familiar with handling "real" dedicated datasets, to process the pictures provided by IGN at `http://forum-micmac.forumprod.com/location-of-sample-data-set` `html` (notice that the svn server supposed to store the pictures seems no longer active, and only the ftp server was reachable to download these datasets whose use is illustrates the documentation [7]).

5.1 Assembling point clouds to display a 3D object

A software named Apero (short for appetizer in French) must be used at least once to model a beer can : this will be our first study case.

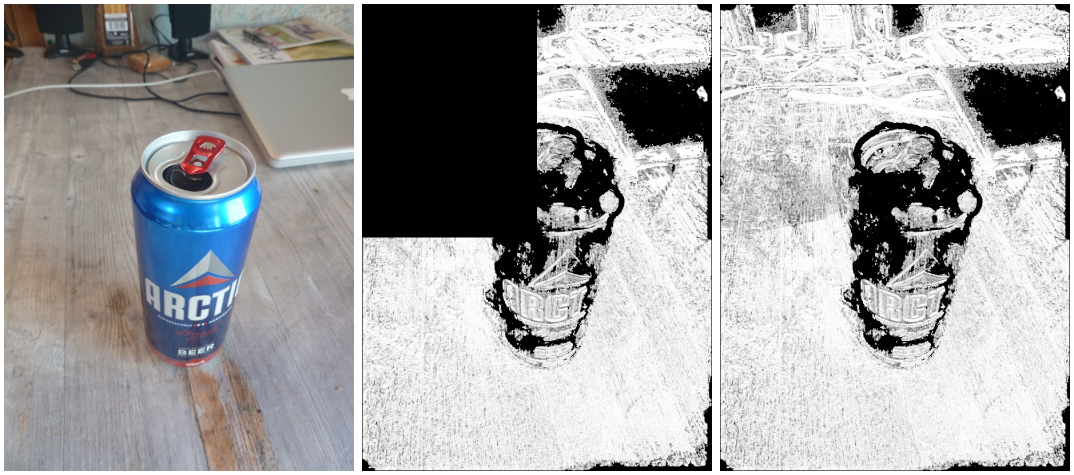


FIGURE 5 – Left : one of the images used to generate the point cloud of the scene displayed in Fig. 6. Center and right : correlation image during the computation, helping monitor the progress of generating the dense point cloud by `Malt`. Right : correlation map once the computation is completed, showing as light colors the areas of strong correlation in which the point cloud is easily computed, and as dark areas the regions in which the computation will fail.

This example provides an opportunity to merge various point clouds acquired with different points of views selected as master images, and analyze the conditions for proper operation of the homologous point identification algorithm. Before considering merging point clouds, it is useful to assess conditions for proper homologous point search and analyze the correlation maps provided by MicMac in the directory including the result of the computation by Malt (`DirMEC=` argument). On the one hand `Corr*` correlation maps indicate the progress of the computation (best viewed by a visualization software which does not prevent writing in the file being shown such as `geeqie`) since the image is assembled while the computation is performed, and on the other hand the white areas indicate regions of strong correlation (high probability of reaching a dense point cloud) and in black the areas of poor correlation (where computing the point cloud will yield noisy or no result at all). The beer can example illustrates the inability of the algorithm to process surfaces without relevant features (computer screen), reflecting surfaces (top side of the can), and obviously transparent surfaces (Fig. 5).

Although the dense point cloud computation by `Malt` following the `GeomImage` model requires a single master image and hence yields a point cloud with the scene shown from a single point of view, this process can be iterated from multiple points of views and hence fill voids by merging complementary point clouds. If the homologous points have *all been identified on the same picture set*, then all these points clouds are generated in an arbitrary yet common coordinate framework. Hence, the resulting point clouds overlap and can be merged using a dedicated software such as CloudCompare. This processing procedure is illustrated in Fig. 6 in which multiple point clouds associated with different points of views are successively added to complete a 3D model of the observed scene. We thus understand how the authors of `https://sites.google.com/site/geomicmac/cavites/tunnel-de-lave` have been able to map a lava tunnel by assembling multiple images : even if not all images overlap while the pictures are being taken, as long as all the homologous points have been identified during the same `Tapioca` processing step by successive overlaps of parts of pictures two by two, all the dense points clouds generated by `Malt` will be defined in a common coordinate system. This result is demonstrated by modelling the entrance to the Fort des Trois Chalets near Besançon (Fig. 7) as a sum of 3 point clouds resulting from processing 5 digital pictures of the tunnel. These results hint at the fact that photogrammetry might be well suited to mapping underground structures, assuming lighting conditions of the walls are constant while pictures are taken.

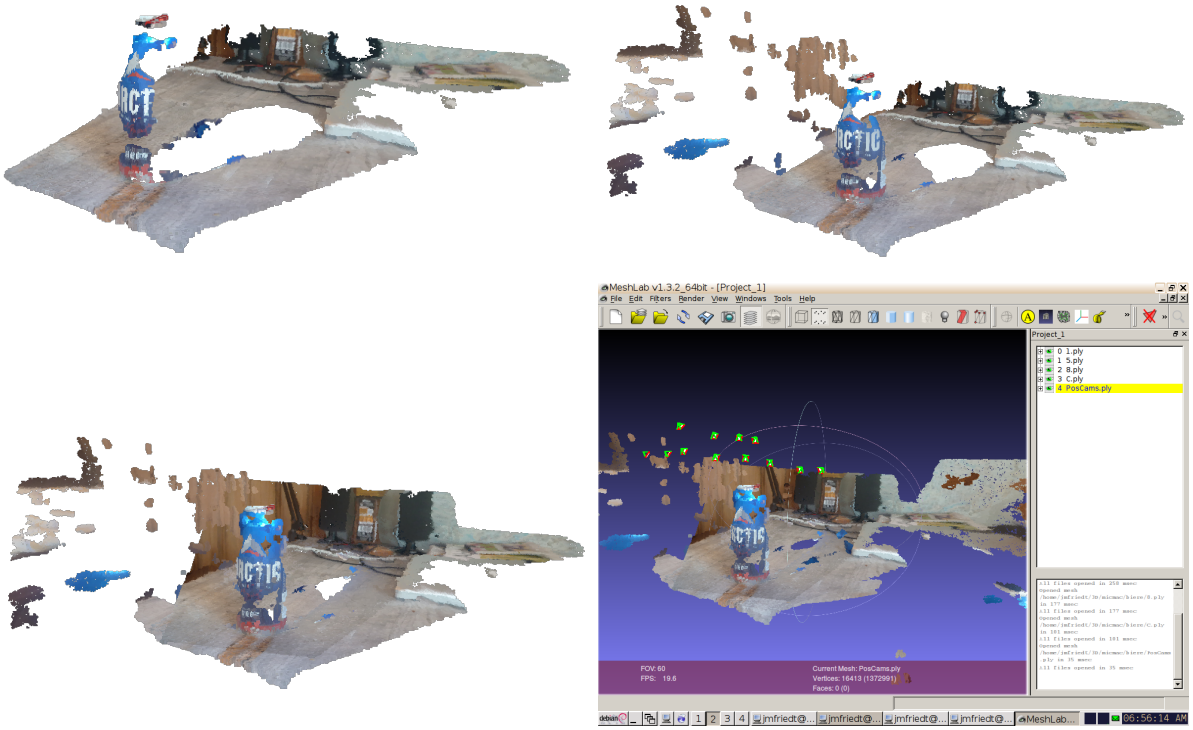


FIGURE 6 – Left to right and top to bottom : sum of 1, 2 and 3 point clouds covering complementary areas of the scene. Each new point cloud results from running the `Malt` and `Nuage2Ply` pair with different `Master` arguments. Notice that the size of the hole in the point cloud behind the can is reduced when each new point cloud is added, which otherwise only complements already existing features of the scene. Bottom right : CloudCompare is used to merge the point clouds, with here the addition of the camera position when pictures were taken.

FIGURE 7 – Point clouds of the entrance of Fort des Trois Chalets near Besançon : tunnels provide a favorable geometry for generating point clouds.

Considering our aim of using poor quality image acquisition systems, the correlation threshold below which the point cloud is not computed is lowered (despite thus reducing the resolution) by adding the following option to `Malt` : `DefCor=0.001`.

5.2 Aerial pictures

In the previous example, we have merged multiple point clouds, each defined from a different master picture for selecting the pixels whose depth is computed during the `Malt` dense point cloud processing. When considering aerial images, this processing sequence is not only lengthy, but it also means that the generated point cloud is only colored on a small subset. The concept of a 3D object is poorly suited to

Digital Elevation Models (DEM), which a better related to a plane over which elevations are extruded. Following such assumptions, we no longer use the `GeomImage` option of `Malt` but `Ortho`, which does not require a master image but only the sequence of aerial pictures being considered. Additionally, a mosaique of orthorectified images can be assembled using the `Tawny` tool in order to generate a large color image for draping the DEM. The processing sequence is hence summarized as

```
mm3d Tapioca MulScale "0(0[5-9]|1[0-9]|2[0-7]{1}).jpg" 300 1000
mm3d Tapas RadialStd "0(0[5-9]|1[0-9]|2[0-7]{1}).jpg" Out=Cal1
mm3d Tapas AutoCal "0(0[5-9]|1[0-9]|2[0-7]{1}).jpg" InCal=Cal1 Out=Ori1
mm3d AperiCloud "0(0[5-9]|1[0-9]|2[0-7]{1}).jpg" Ori1 Out=PosCams.ply
mm3d Malt Ortho "0(0[5-9]|1[0-9]|2[0-7]{1}).jpg" Ori1 "DirMEC=Resultats3" UseTA=1 ZoomF=4 ZoomI=32 Purge=true
mm3d Tawny Ortho-Resultats3/
Nuage2Ply Resultats3/NuageImProf_STD-MALT_Etape_6.xml Attr="Ortho-Resultats3/Ortho-Eg-Test-Redr.tif"
```

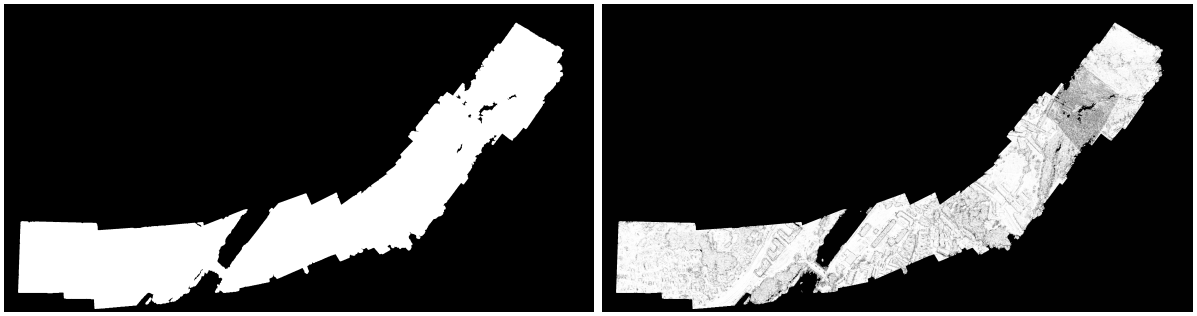


FIGURE 8 – Left : mask identified by `Malt` during the assembly of aerial pictures over the average elevation plane. Altitudes are only computed in white areas. Right : correlation level map during the computation of the dense point cloud. Notice the lack of correlation over the river.

The picture set assembled to generate the digital elevation model along a strip located from the Besançon castle to the Doubs river was acquired from a motorized ultralight plane fitted with a digital reflex camera positioned for a vertical view of the scene below. The mask (Fig. 8, left) displays, in the average plane of the DEM, the pixels (white) for which the elevation will be computed by processing the pictures. The correlation level map (Fig. 8, right) exhibits lighter colors for well defined features visible on the ground : urban environments are most favorable, yielding high success rates, while the Doubs river yields no correlation at all since water does not exhibit reproducible features as successive pictures are taken. Such conclusion are validated by comparing the mosaique of the images corrected from topography related deformations and oblique view (orthoimages : Fig. 9, gauche) with the colored point cloud (Fig. 9, right) : the areas in which the correlation was inexistent do not contain any point in the 3D cloud model. The third dimension is best observed in an oblique view representation demonstrating the consistency of the computed elevation map (Fig. 10). Instead of the `Ortho` processing model, the `UrbanMNE` mode [7, p.63, sec. 3.12.1] is tuned by reducing the convolution filter width (from 5×5 to 3×3 pixels) to provide sharper results, but its default behavior is to *not* compute orthorectified image maps. The latter result is nevertheless achieved by adding the `LrOr=true HrOr=true` options to the `Malt UrbanMNE` command but in this example, we have not observed any significant difference between the `Ortho` or `UrbanMNE` model point cloud results.

We will later see (section 9) that this computation can be performed in an absolute coordinate system if the position of the camera is known when each picture is taken (for example by synchronizing picture acquisition with a GPS receiver). Under such circumstances, two files in the `Ortho-` provide the orthoimage (Fig. 9) position information : `MTDOrtho.xml` includes the origin and size of each pixel in a human readable format, and `MTDOrtho.tfw` provides these same information in the standardized format of geolocated TIF images.

5.3 Picture ordering

Searching for homologous points between pictures is a key step for later identifying lens optical properties and the position of the camera as each picture is taken : this is the very first processing

FIGURE 9 – Left : mosaique of the orthorectified images generated by `Tawny` for defining the color of the pixels in the point cloud (`Ortho-Eg-Test-Redr.tif` file in the `Ortho-` directory provided as argument to `Tawny`). Right : the colored point cloud, azimuthal view. Notice the lack of samples over the river which is associated with a lack of correlation between images.

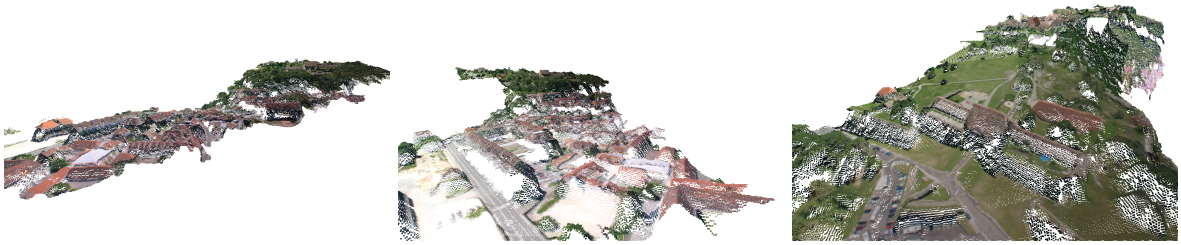


FIGURE 10 – Same point cloud as seen in Fig. 9, but this time displayed in an oblique view configuration, demonstrating the altitude information associated with each pixel.

step when running `Tapioca`. The processing option `MulScale` is valid under most circumstances. One particular case was met in which this method fails : when taking pictures along a street we wanted to map, the software mistakenly associates features on walls belonging to different buildings (Fig. 11). The consequence is an erroneous positioning of the cameras which are not longer situated along the street in the model, so that the later steps of the processing sequence are doomed to fail (hence the need to always validate the camera position by using `AperiCloud`). The solution is to ask `Tapioca` to use the `Line` method which reduces the search of homologous points *only* to the pictures indexed $\pm N$ in the picture sequence, with N provided as the last command line argument. Not only do we reduce computation time of the first processing step, but most importantly we prevent points on different buildings from being associated as common features. A similar issue was met for a chapel whose adjacent walls looked similar on the pictures, or the sides of a tower on the Vauban fortifications in Besançon (section 9.2).

As a last resort, if the automated match of pictures representing common features fails, the last solution is to provide a list of picture pairs in a file processed thanks to the `File` option of `Tapioca` in an XML format such as

```
<?xml version="1.0" ?>
<SauvegardeNamedRel>
     <Cple>img1.JPG img2.JPG</Cple>
     <Cple>img1.JPG img3.JPG</Cple>
...
</SauvegardeNamedRel>
```

5.4 Recovering images from a movie

Most readers are probably not lucky enough to fly an ultralight plane or a helicopter to acquire dedicated sets of pictures. One possible source of data is to extract aerial views from documentaries and process such a dataset. We have focused on the documentaries produced by Sylvain Augier (l'Europe

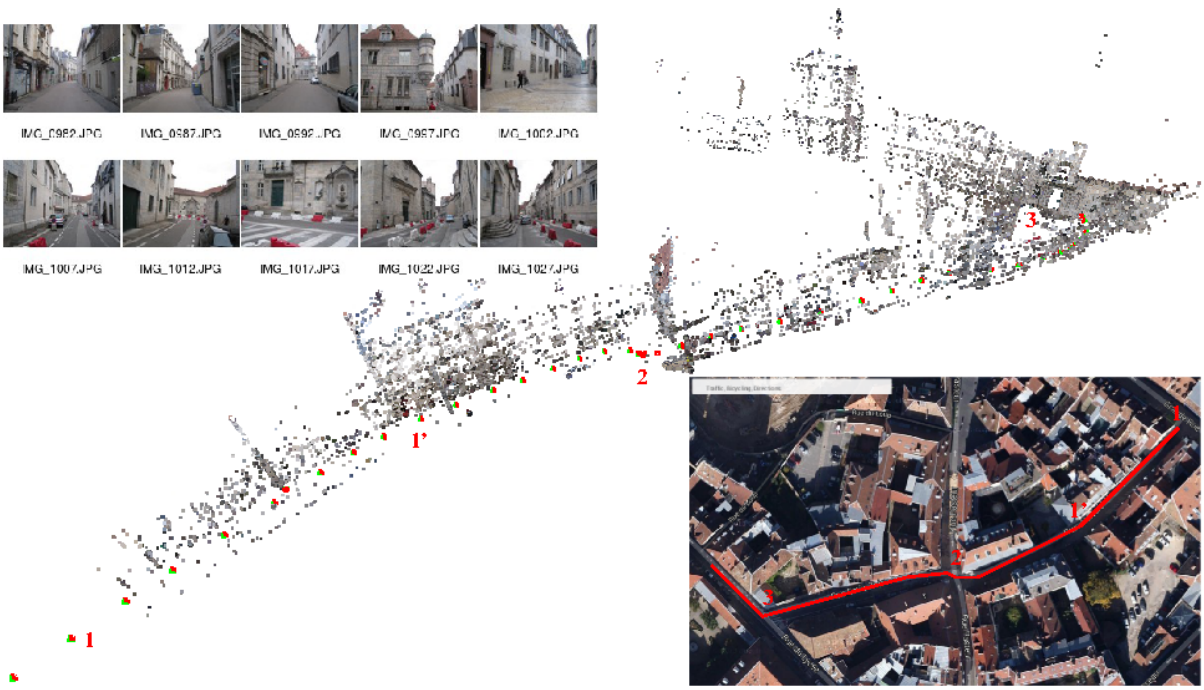


FIGURE 11 – Pictures taken along a street : proper positioning of the cameras is only possible when using the `Tapioca Line 1500 3` command, indicating that the homologous point search must be restricted to the ±3 pictures around the picture being considered. Insert : path followed as seen in the aerial view of Google Maps, with four reference points indicated along the track. MicMac was unable to properly analyze the last right-angle corner after reference point 3 : the positions of the camera after this point are incorrect. With a range of 5 pictures instead of 3 when searching for homologous points, the right angle is identified but the scale in the new direction is incorrect. Top insert : some of the 60 pictures acquired (compact digital camera Canon PowerShot A2200).

vue du Ciel [11], l'Île de France vue du ciel [12]) or by the Arte television channel (Les Alpes vues du Ciel [13]. MicMac is so robust that we can fill the fields associated with each image extracted from the movies with nearly any random value, and `Tapas` will correct these parameters during the lens property and camera position identification steps. A zoom on a monument during the movie acquisition dooms the computation to failure, which otherwise yields good results if the cameraman did not change the magnification as the movie was shot (Fig. 12).

Extracting individual images from the movie is performed using `mplayer` by selecting as video output jpeg files with the best possible quality : `mplayer -vo jpeg:quality=100 film.avi`. In order to start playing the movie with the part we are interested in, the `-vo` is prefixed with the `-ss start` option, with `start` the date (in seconds) of the beginning of the interesting part. Obviously, at a 25 frame/second rate, the resulting number of images is huge and we only keep one in every 10 images, or one picture taken every 0.4 s. Processing too many pictures is useless, and one should favor diversity in the points of view of the targeted site in order to generate a good quality point cloud. An EXIF header is mandatory for MicMac to process images [14], even if the actual content is meaningless : we thus simply copy the EXIF header of any picture name `ref` taken by a digital camera (JPEG format) without caring whether the image size or focal length match the actual image parameters from the movie : `exiftool -TagsFromFile ref *.jpg`.

Perspectives of using documentaries available on the internet are hence nearly unlimited (Fig. 13),

11. `http://www.editionsmontparnasse.fr/p1001/L-Europe-vue-du-ciel-filmee-par-Sylvain-Augier-DVD`
12. `http://www.editionsmontparnasse.fr/p1202/Paris-Ile-de-France-vus-du-ciel-filmes-par-Sylvain-Augier-DVD`
13. `http://www.arte.tv/guide/fr/044681-002/les-alpes-vues-du-ciel` or `http://www.arte.tv/guide/fr/044681-004/les-alpes-vues-du-ciel`
14. we do not use here the ability to provide MicMac with the information missing in the EXIF header by filling the `MicMac-LocalChantierDescripteur.xml` file in the directory in which all the pictures are located

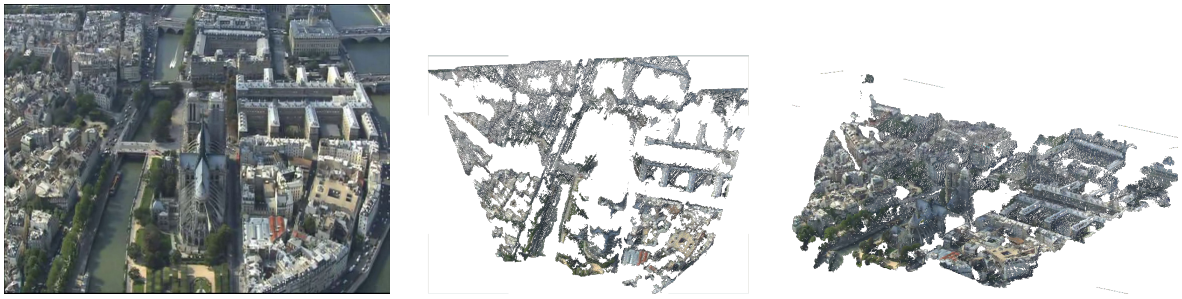


FIGURE 12 – Model of the Paris church Notre Dame and Île de la Cité obtained by processing images extracted from the Paris Vu du Ciel documentary. Left : one of the pictures extracted from the DVD for processing. Middle : azimuthal view of the point cloud, on which Île de la Cité and the largest buildings are clearly visible. Right : oblique view of the same point cloud, allowing the visualization of the two protruding towers of the church.

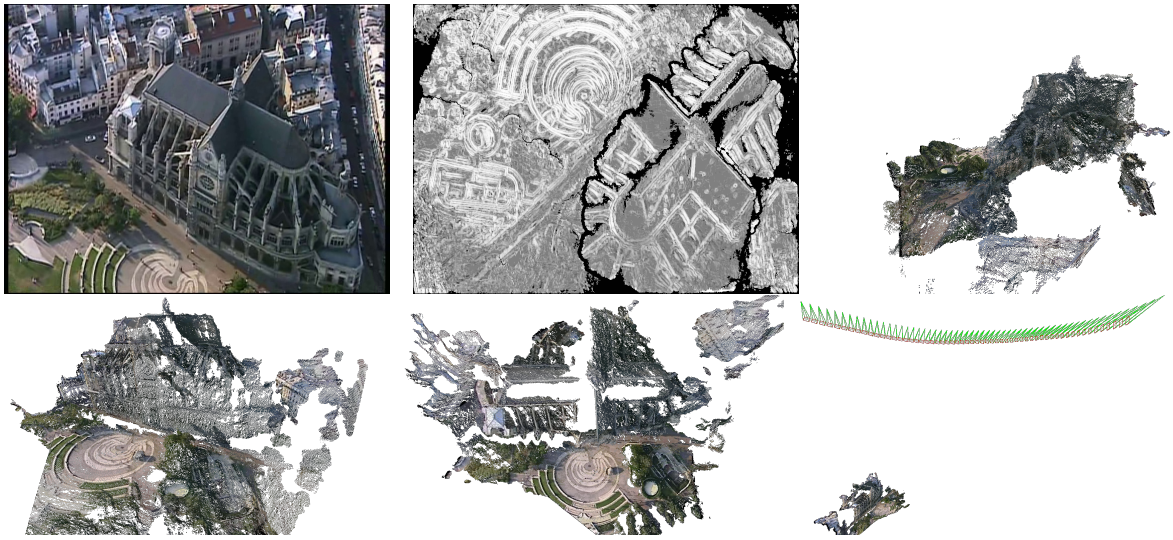


FIGURE 13 – Top, from left to right : one of the 58 images extracted from the Saint Eustache church part of the "Paris Vu du Ciel" documentary ; correlation map hinting at the ability to process this dataset despite the poor movie resolution (768×576 pixels) ; one of the three views of the point cloud representing the church model. Bottom : another two views (from the parvis and azimuthal projection) of the point cloud representing the church ; and right : the position of the helicopter when the movie was shot. Notice the elongated shape of the green triangles, indicating the use of the telephoto lens, a usually poor selection for photogrammetric processing of the images. A quantitative analysis of the relative heights indicates that the aircraft was flying at an altitude equal to 6.7 times the height of the church. Assuming a vault height (`http://en.wikipedia.org/wiki/Saint-Eustache,_Paris`) of 33.46 m, then the flight altitude is about 225 m, above the legal 200 m threshold imposed since 1998 to reduce noise due to aircrafts, and below the 350 m accessible to planes.

but compliance with shooting constraints is usually not met on amateur documents available for example on YouTube (users of GoPro cameras – whose fixed focal length is most appropriate for our purpose – often attach the camera to a helmet which keep on moving and focusing on different fields of view). Shots taken during extreme sports events such as wingsuit are thus hardly usable for a quantitative analysis of the pictures, while slower activities such as gliding might provide a useful dataset, assuming only fixed magnification sequences are used (Fig. 14).

The model of Briançon required using a `GeomImage` model since the plane flight path was circling around the city while the movie was being shot. Another example from the second volume of "La France

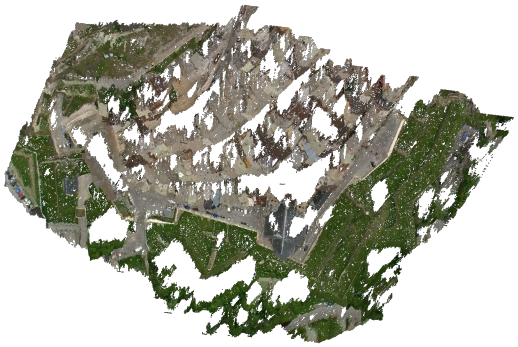


FIGURE 14 – Model of Briançon in an azimuthal projection generated from oblique view pictures taken from the "Les Alpes Vues du Ciel" documentary.

vue du Ciel" [15] provides a model of Sète (Fig. 15) extracted from a long sequence in the movie processed by MicMac following an `Ortho` model when running `Malt` and corrected for topography effects, despite the poor quality of the DVD.

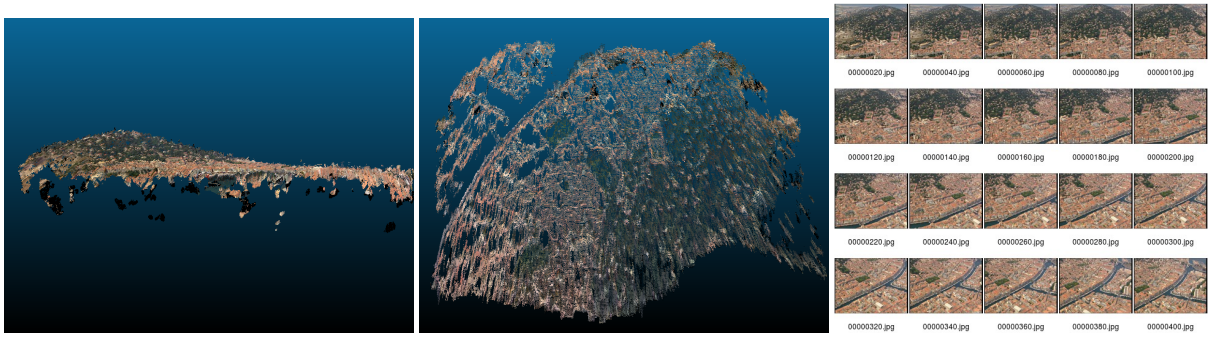


FIGURE 15 – Sète, oblique view demonstrating the topography generated from the pictures extracted from the movie, and azimuthal view. Right : some of the images taken from the movie used to generate the point cloud.

6 Second case ... aerial pictures

A small drone sold as a toy, the RC System Space Q4, is provided with an embedded camera and costs a bit over 100 euros. This toy is demonstrated to become an excellent source of aerial views despite the poor camera resolution and optics quality.

Indeed, the fixed lens and the lack of zoom capability guarantee that the images extracted from the movie recorded while the drone was flying – again using `mplayer -vo jpeg` – meet the requirements needed for processing with MicMac (Fig. 16).

Here too, the missing EXIF header of the pictures extracted from the movie is filled as explained in the previous section. A fascinating consequence of positioning the camera during the image processing steps is that the flight path in space is reconstructed (Fig. 17). Such results might become useful when analyzing flight parameters in order to identify command laws while designing a drone : while an Inertial Measurement Unit (IMU) only provides the second derivate (acceleration) of the position and the first derivate of the orientation (gyrometer), here we have the opportunity to be informed of the position and orientation (angles) of the aircraft.

15. `http://www.editionsmontparnasse.fr/p882/La-France-vue-du-ciel-filmee-par-Sylvain-Augier-DVD`

FIGURE 16 – Indoor flight of the drone fitted with its camera

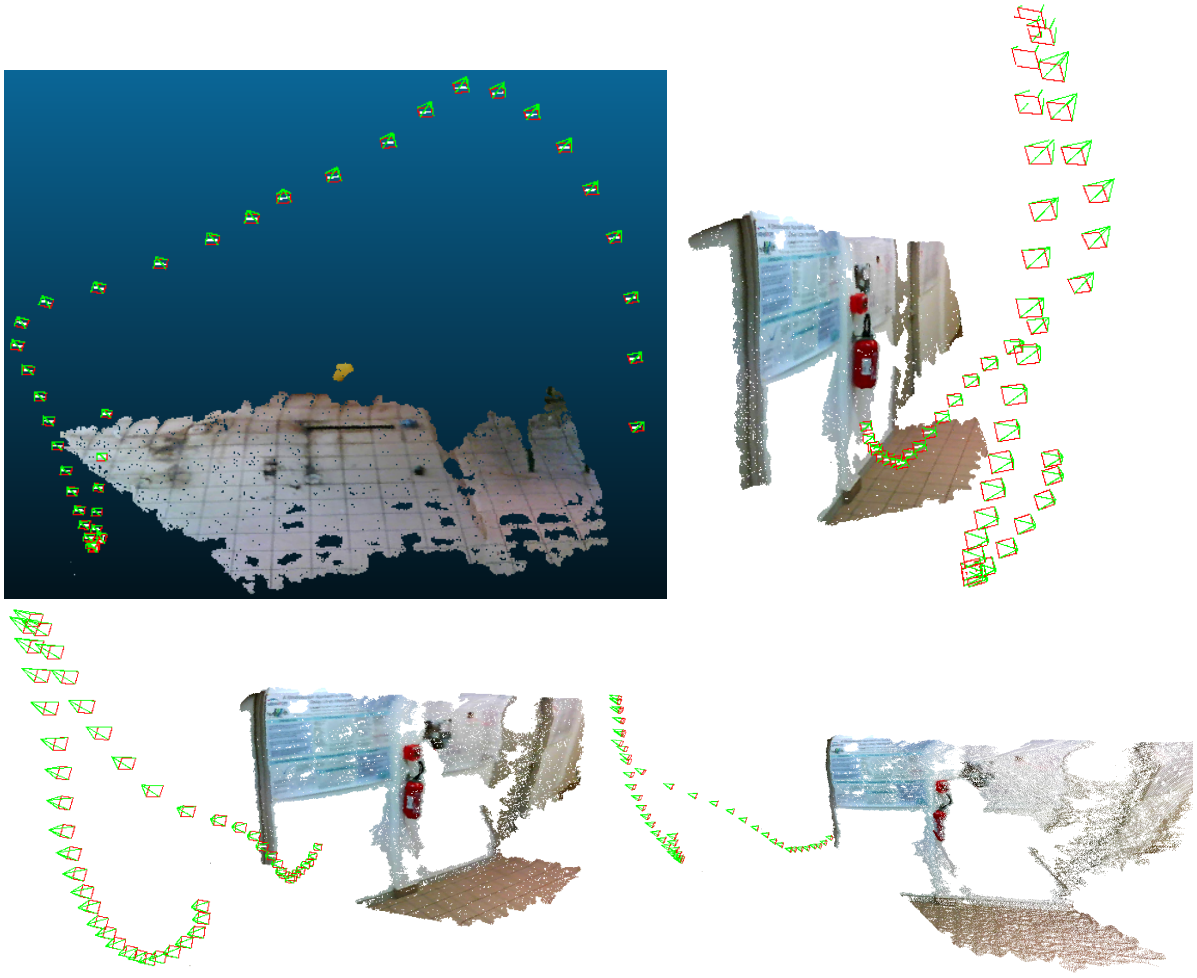


FIGURE 17 – Top right and bottom : 3D map of a laboratory hall lines with posters generated from the pictures acquired by a drone. The flight path reconstruction is as impressive as the generated model of the environment in which the drone flew. Top left : trajectory reconstruction during a flight in an empty paved room. Notice how skilled the pilot is when narrowly preventing the crash on the left part of the trajectory !

7 Locating the camera, and comparison with the GPS position

Having seen that the camera position could be identified using the recorded images, one might wonder how consistent the position deduced from the MicMac processing is with respect to the Global Positioning System (GPS) considered as the reference.

FIGURE 18 – Left to right and top to bottom : one of the images shot from the train linking Besançon to Dijon with a Samsung-S3 mobile phone in order to generate a 3D view of the scene ; 3D reconstruction of the building from the digital picture, including the position of the camera when each picture was shot ; azimuthal view of the point cloud ; positioning of the point cloud in Google Earth. The green and red arrows guide the eye towards the parts of the azimuthal view inserted in the background aerial picture. Notice the excellent correlation between the camera position and the right rail of the track.

The agreement between the position computed by image processing by MicMac and the reference position deduced either from the knowledge of the path followed when taking the pictures or the GPS position is excellent. A first example illustrating this aspect results from a set of images of a building along the tracks shot from the train linking Besançon to Dijon, near the destination train station. The azimuthal view of the resulting point cloud was scaled and positioned over an aerial map taken from Google Earth. The building point cloud includes the camera position as the train was moving, which is hence also positioned on the Google Earth background image. The consistency of the camera position is excellent, located over the right rail of the track, agreeing with the position of the photographer in the car (Fig. 18). Another example of the excellent agreement between the positioning resulting from the MicMac computation and the GPS record of the camera position is provided in [11].

Both comparison examples are purely geometrical superposition of two datasets, here achieved using the Gimp, one the one hand the azimuthal projection of the point cloud and on the other hand the aerial view from Google Maps. Once the scaling, translation and rotation properly tuned (in our case by iterative trial and error) so the buildings overlap, no degree of freedom is left for placing the cameras on the tracks : the agreement is the result of the proper result generated by the algorithms implemented in

MicMac (`Tapas`) for identifying the optical properties and position of the camera in order to generate the point cloud. Having reached this conclusion, we can consider the problem the other way around, and assume we know either the position of some reference point on the picture (ground control points – GCP), or when shooting the pictures, and hence convert the arbitrary coordinate system in which the point cloud is located to an absolute framework physically relevant. Such a strategy will be followed in the next two section, first by considering GCP position known and visible on multiple pictures, and then by exploiting the known position of the camera when the images are shot.

8 From qualitative to quantitative : ground control points

One way of generating a quantitative point cloud, with dimensions expressed in known units (centimeters, meters ...) instead of pixels, is to provide the 3D coordinates of some known reference points visible on all (or most) pictures. All points do not need to be visible on all pictures, but maximizing the overlap obviously yields best results. This method introduces no assumption concerning the optical setup or the instrument used to take the images but only relates the 2D position of the GCP on the picture with the 3D position in space. How correct the result is depends on how careful we are in locating the GCP, but even without being precise the result is impressively accurate. Our tests show that only providing distances on the ground on which an object is located is enough to identify the lengths along the third dimension (height above ground) with an accuracy equal to the resolution with which the object was measured (about ± 5 mm on a 12 cm high object).

Poor optical quality does not prevent image processing. A basic mobile smarthphone is enough, as shown in the following example using images acquired with a Samsung S3 mobile phone. The EXIF header lack the information of the 35-mm equivalent focal length of the lens and this value must be added with a more or less randomly selected value (and motivated by data gathered on the web) : `exiv2 -m set_exif.jmf *.jpg` where the command file `set_exif.jmf` is filled with a single line `set Exif.Photo.FocalLengthIn35mmFilm 21` since 21 mm seems to be a commonly accepted value for the widest zoom setting of this phone. Changing this setting in the EXIF header slightly moves the point cloud in space (for example by replacing 21 mm with 40 mm) but does not affect the accuracy of the object height.

The processing steps are fully provided here despite the similar first instruction with the previous examples, in order to identify when the GCP information is inserted for converting the arbitrary coordinate system towards the GCP coordinate system by using `GCPBascule` :

```
mm3d Tapioca MulScale ".*jpg" 300 1500
mm3d Tapas RadialStd ".*jpg" Out=Init
mm3d AperiCloud ".*jpg" Init
mm3d GCPBascule ".*jpg" Init Ground Ground-Pts3D.xml GroundMeasure.xml

# weighted compensation beween camera position and GCP with Campari in Final
mm3d Campari ".*jpg" Ground Final GCP=[Ground-Pts3D.xml,0.1,GroundMeasure.xml,0.5]

# comparison of the result without weighting ...
mm3d Malt GeomImage ".*jpg" Ground Master=20140518_110653.jpg DirMEC=Results ZoomF=4 ZoomI=32 Purge=true
mm3d Nuage2Ply Results/NuageImProf_STD-MALT_Etape_6.xml Attr=20140518_110653Zoom4.JPG

# ... and with weighting ...
mm3d Malt GeomImage ".*jpg" Final Master=20140518_110653.jpg DirMEC=Campares ZoomF=4 ZoomI=32 Purge=true
mm3d Nuage2Ply Campares/NuageImProf_STD-MALT_Etape_6.xml Attr=20140518_110653Zoom4.JPG
```

The key step is calling `GCPBascule` with an input argument being the directory with the initial orientation of the camera (`Ori-Init`) and exploiting the 3D position of the GCP (here provided in cm units in the `Ground-Pts3D.xml`file) as well as the position (in pixels) of these same GCP in each picture, as provided in `GroundMeasure.xml`. The format of these files is simple and inspired from the `Dico-Appuis.xml` (3D position of GCPs) and `Mesure-Appuis.xml` (2D position of GCPs in each picture) found in the gravel example available at à `http://logiciels.ign.fr/?-Micmac,3-` : each GCP is given a name (`NamePt` tag) and a position (`Pt` tag) in `Ground-Pts3D.xml`. For each picture (`NameIm` tag) found in `GroundMeasure.xml`, we provide the name of the GCP (`OneMesureAF1I` tag) and its pixel coordinates on the picture (`PtIm` tag). Once the coordinate framework changeover is completed, the dense cloud point is generated in the new framework and will allow, using CloudCompare, to locate the position of

various points of the modeled structure (the icon depicting a target is used to request the properties of a point, including its coordinates in space). The result of these processing steps on a teddy bear is shown on Fig. 19, and was performed in another context in [11].

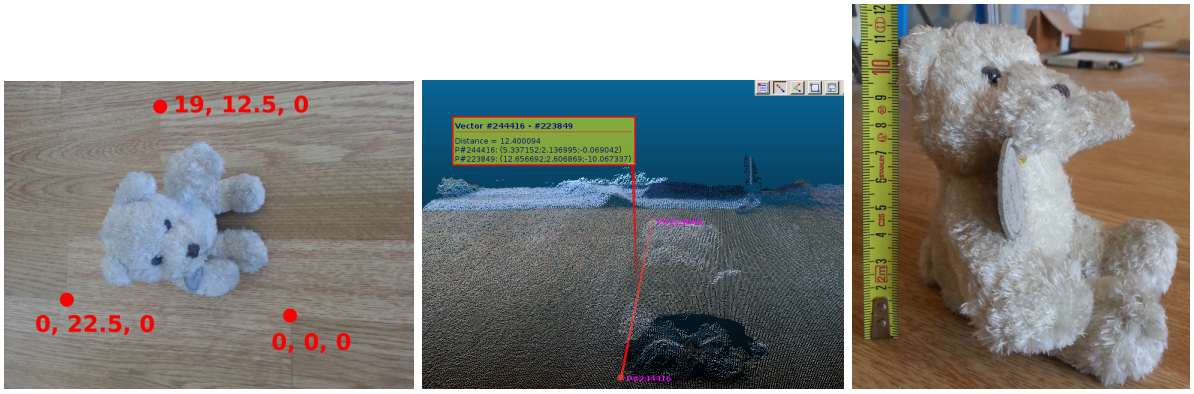


FIGURE 19 – Left : GCP coordinates – in cm – of selected corners of the slats of a floor, visible from most of the 6 pictures acquired of the teddy bear using a mobile phone. Middle : point cloud generated and measurement in CloudCompare of the height of one ear of the toy, considering that all GCP altitudes were set to 0 and hence the dimensions along the third dimension are exclusively the result of the photogrammetric processing. Right : measurement of the height of this same ear. The error is about 10% on this example which was not compensated using `Campari`.

This analysis is completed by adding another functionality provided by MicMac in order to weight the GCP position – necessarily flawed – with the expected position resulting from the photogrammetric computation. Having completed the switching from the arbitrary coordinate system (`Ori-Init` orientation directory) to the GCP coordinate system (`Ori-Ground` orientation directory), we ask MicMac to compensate for errors when measuring the coordinates of the GCPs, based on the information it gathered from the scene : this task is performed by `Campari` with the GCP coordinate orientation directory, the XML files with the 3D and 2D coordinates of the GCPs, and the relative weights (between GCP and camera positions extracted from the photogrammetric processing) as arguments. We have selected to perform or not perform this weighting operation : Fig. 20 shows the difference between the two resulting point clouds, as computed by CloudCompare (functionality provided by the 8th icon from the left, after loading and selecting the two point clouds located in the `Campares` and `Results` directories).

This way of working is tedious because it requires locating the GCPs on each picture and to fill the position information. Tools are provided to ease this task – `SaisieAppuisInit` and `SaisieAppuisPredic` – but their use is beyond the topic of this presentation.

9 From qualitative to quantitative : exploiting the camera position when the pictures were taken

An alternative solution to identifying GCP positions on each picture is to provide the position of the instrument used to shoot the pictures and to deduce the scale and orientation of the scene from this input information. This approach is best suited for large scenes, in which GCP positions are not necessarily known with enough accuracy but a GPS receiver next to the digital camera provides the position with enough accuracy for the task at hand [12] [16].

9.1 Spreading a sphere on a plane

The problem with using GPS coordinates lies in the conversion from a spherical coordinate system – latitude and longitude in degrees – to a Cartesian coordinate system using a local approximation

16. Notice that an IGN laboratory has developed a GPS datalogger, `loemi.recherche.ign.fr/pdf/brochureGeocube1.pdf`, providing centimeter resolution in static measurement conditions !

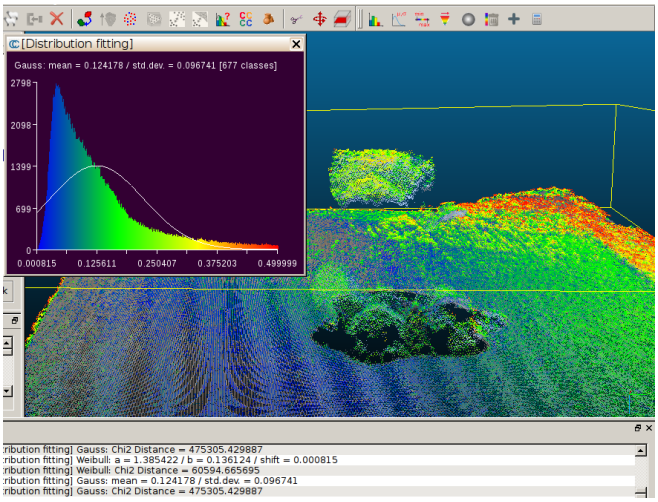


FIGURE 20 – Distance between two cloud points – with and without applying the `Campari` compensation – displayed on a scale ranging from 0 (blue) to 0.5 cm (red). A gaussian fit of doubtful quality hints at an average difference of 0.12 cm, with an error that reaches 3 mm at the top of the head of the toy.

of a sphere as a plane. Similar to the pirate who has hidden his treasure map, the geographer rather talks in steps (or meters in the modern naming convention) to the north or the east rather than in angular degrees. In order to georeference camera positions in a coordinate framework graduated in meters, we must convert the GPS position of the camera recorded when each picture is shot as discussed in [12] (latitude/longitude) to abscissa/ordinate/altitude in meters. Although MicMac links a library dedicated to coordinate transformations (`ChSys` option of `GCPConvert`) – an excessively complex task when accounting for the non-spherical shape of the Earth in order to reach meter accuracy – we have translated from the spherical coordinate system (WGS84 framework used by all GPS receivers) to the projected coordinate system (Cartesian) by using QGis (`www.qgis.org/`).

The procedure is as follows :

1. identify, by matching dates and accounting for clock difference between the GPS time and digital camera time, the position in the WGS84 spherical framework (latitude/longitude) of each picture. The GPS included in some of the digital cameras (at least the one found in the Panasonic TZ10, to be avoided for any application other than showing off) provide an insufficient number of decimals to be usable and an using an external GPS receiver is mandatory,
2. save in ASCII format a file including the list of latitude/longitude/altitude of each picture,
3. load this file in QGis by using the icon shaped like a coma (Fig. 21),
4. save this same file by selecting the projection mode, depending on the geographical area being considered. Indeed, since the Earth is not spherical, local fitting of the potato-shape is needed to find the tangent plane in order to project the spherical coordinates to a Cartesian coordinate system [13, pp.228-241]. WGS84 is a spherical coordinate system (degrees) which is projected to the WGS84/UTM`region` (where `region` indicates the considered longitude) to find the tangent plane best suited to each region – for France, we consider region 31 (hence UTM31N).
5. use the resulting file with MicMac when running the coordinate system conversion provided with `OriConvert`.

At the end of these processing steps, the file containing the coordinates of the camera when each picture was shot is converted from

```
Y X Z
49.20993499999999 3.085168333333333 124.3
49.20936 3.084938333333333 123
49.20967333333333 3.08532 121.6
...
```

to

```
X,Y,Y,X,Z
506203.20328651543241,5450797.176053959876299,49.209935,3.08516833333333,124.3
506186.52307022450259,5450733.23488206975162,49.20936,3.08493833333333,123
506214.282665384875145,5450768.099197551608086,49.2096733333333,3.08532,121.6
...
```

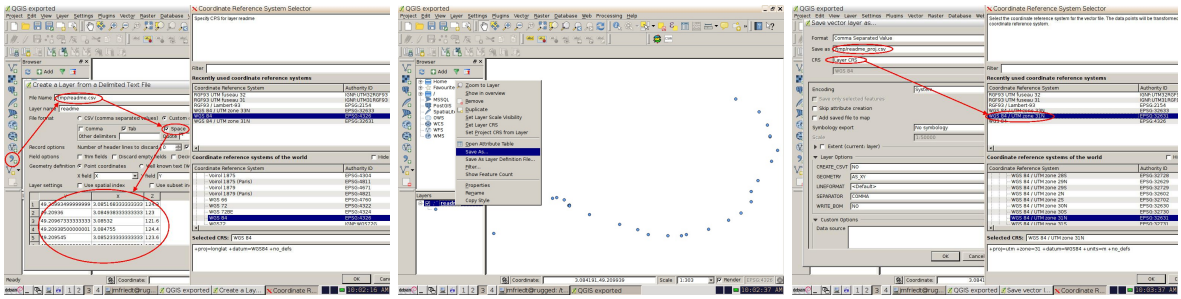


FIGURE 21 – Operation sequence when using QGis to project the GPS coordinates of the locations at which the pictures were shot.

Practically, the initial processing sequence is

```
mm3d OriConvert OriTxtInFile readme.gpsutm33Ncut Nav-RTL MTD1=1 NameCple=FileImagesNeighbour.xml
Tapioca File FileImagesNeighbour.xml -1
mm3d Tapas RadialStd ".*JPG" Out=Calib
mm3d Tapas AutoCal ".*JPG" InCal=Calib Out=All-Rel
mm3d AperiCloud ".*JPG" All-Rel Out=PosCams-Rel.ply
CenterBascule ".*JPG" All-Rel Nav-RTL Abs
mm3d AperiCloud ".*JPG" Abs Out=PosCams-Abs.ply
```

here we help `Tapioca` by using the GPS position of the camera when the pictures were shot as it looks for the homologous points between images closely located. Alternatively, we can search for all homologous points between all picture pairs and identify the camera positions in the arbitrary coordinate system used by `Tapioca` and `Tapas`, before converting to the absolute GPS coordinate by using `CenterBascule` and complete, in this new framework, the generation of the dense point cloud (`Malt` on the orientation directory `Abs`). The two point clouds viewed using `CloudCompare`, `PosCams-Rel.ply` and `PosCams-Abs.ply`, validate these processing steps as seen by requesting the characteristics of a few points (icon exhibiting a target) and verifying that the arbitrary coordinate framework was indeed converted to the GPS absolute coordinate system.

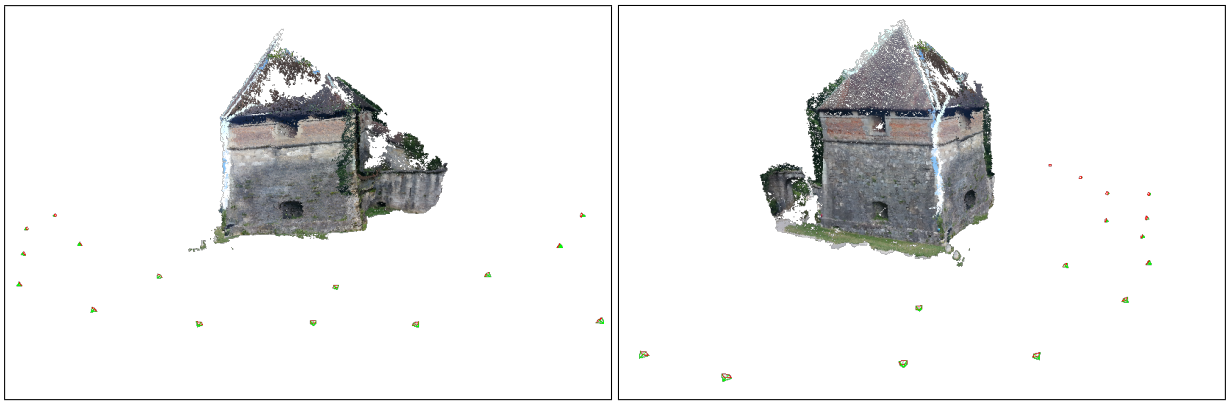


FIGURE 22 – A tower along the Besançon battlements, also displaying the locations at which pictures were shot using a smartphone fitted with a GPS receiver.

The first line (`OriConvert`) converts the input file (called `readme.gpsutm33Ncut`) including

```
#F=N X Y Z
#
#image          longitude               latitude        altitude
P1130740.JPG    506173.166790323157329  5450735.999143296852708 124.4
P1130733.JPG    506148.26461441826541   5450750.979183392599225 126.5
P1130730.JPG    506128.860635785094928  5450734.096877036616206 128.9
```

to a file which can be processed by MicMac. The first line of this file is processed to define the text file content format [7, section 12.3] : here the first character indicates that any line beginning with # will be considered as a comment, and then the following columns are respectively the name of the picture, abscissa, ordinate and altitude. An additional keyword, `S`, could indicate that a useless additional field is included in the text file (which will not be analyzed by `GCPConvert` or `OriConvert`). Specifically, the XML file includes the image pairs so that `Tapioca` doesn't have to look at all possible combinations but uses the GPS coordinates to identify which images were shot from nearby locations. The next steps are similar to those already described : identification of the optical properties of the camera used to shoot the pictures and rough point cloud including the camera position. The novel processing step lies in calling `CenterBascule` to convert the arbitrary coordinate system to the absolute framework. Finally, `Malt` completes the computation of the dense point cloud by relying on the parameters provided not by the `All-Rel` orientation directory but on the information lying in the orientation directory that was generated by `CenterBascule`, namely `Nav-RTL`.

9.2 Inserting the point cloud in a geographic information system

These processing steps are used to model a tower which is part of the Vauban fortifications in Besançon. The reader can check the geographic environment by looking for the coordinate 47.2308N, 6.0186E in Google Maps (entering these coordinates in the search bar).

Starting from the dataset considered earlier, we were careful to shoot these pictures using a mobile phone fitted with a GPS receiver and running OSMTracker [17]. This application is well suited to generate a GPX formatted log file, very simple to analyze, in which the pictures shot using this application are marked as waypoints (`wpt` tag) providing their position (WGS84 framework). After extracting these informations, we end up with a file including

```
N       Y       X       Z
10.jpg 47.23082994114249 6.019135079959552 302.07939594541966
11.jpg 47.23082709250457 6.0189871931638095 300.11863592268435
12.jpg 47.230800730543464 6.018898687257535 298.2475180903055
..
```

which is projected to the WGS84/UTM31N framework by using QGis to yield

```
X,Y,N,Y,X,Z
728532.941096769180149,5235237.991666674613953,10.jpg,47.2308299411425,6.01913507995955,302.07939594542
728521.75988510530442,5235237.241747501306236,11.jpg,47.2308270925046,6.01898719316381,300.118635922684
728515.174332492286339,5235234.053077357821167,12.jpg,47.2308007305435,6.01889868725754,298.247518090306
728510.913302066153847,5235230.523033961653709,13.jpg,47.2307704923071,6.01884067241403,296.38010190175
...
```

The first two columns provide the coordinates in a UTM framework, the next two are spherical coordinates, and the last column is the altitude. If the next MicMac computations are performed on this set of coordinates, the scene represented by the point cloud is tilted due to the uncertainty on the altitude : while a few meter errors are acceptable in this application in XY, the huge relative error on the altitude induces a significant error on the final scene orientation. Since we were on a towpath whole altitude does not significantly vary around the tower, we have replaced all values of the last column (altitude) with a constant value for all lines, selected at the round value of 300 m (Fig. 23). We have also moved the X and Y axis in order to remove the first digits which induce rounding errors during the computation (we subtract 728000 to X and 5235000 to all Y values : these offsets must be remembered since we will have to add them back at the end of the point cloud computation during the georeferencing process). The final document, ready to be used in the MicMac processing chain, includes the camera coordinates for each shot (file named `position_UTM33_cut_Zcst.csv`) :

17. `https://code.google.com/p/osmtracker-android/`

```
#F=N X Y Z
10.jpg 532.941096769180149 237.991666674613953 300.
11.jpg 521.75988510530442 237.241747501306236 300.
12.jpg 515.174332492286339 234.053077357821167 300.
13.jpg 510.913302066153847 230.523033961653709 300.
...
```

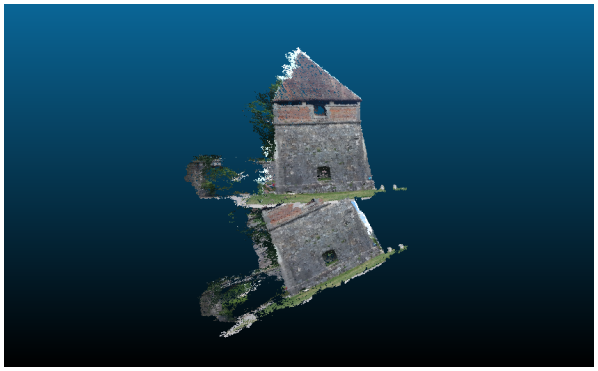


FIGURE 23 – Effect of the camera position altitude uncertainty on the point cloud orientation. The tilted tower results from using raw GPS measurements exhibiting ±5 m altitude variations. The "vertical" tower is the result of the same computation, but this time with all altitudes set to the same value for all shots.

The processing sequence thus ends up being

```
mm3d Tapioca MulScale "(1[0-9]|2[0-4]).jpg" 300 1500
mm3d Tapas RadialStd "(1[0-9]|2[0-4]).jpg" Out=Init
mm3d Tapas AutoCal "(1[0-9]|2[0-4]).jpg"  InCal=Init Out=Init1
mm3d AperiCloud "(1[0-9]|2[0-4]).jpg" Init1

mm3d OriConvert OriTxtInFile position_UTM33_cut_Zcst.csv jmfgps
mm3d CenterBascule "(1[0-9]|2[0-4]).jpg" Ori-Init1 Ori-jmfgps Abs0
mm3d Campari "(1[0-9]|2[0-4]).jpg" Abs0 Abs1 EmGPS=[jmfgps,0.1]]

mm3d Malt GeomImage "(1[5-9]).jpg" Abs1 Master=16.jpg DirMEC=Result1 ZoomF=4 ZoomI=32 Purge=true DefCor=0.001
mm3d Nuage2Ply Result1/NuageImProf_STD-MALT_Etape_6.xml Attr=16.jpg RatioAttrCarte=4

mm3d Malt GeomImage "(1[1-3]|2[0-2]).jpg" Abs1 Master=21.jpg DirMEC=Result2 UseTA=1 ZoomF=4 ZoomI=32 \
      Purge=true DefCor=0.001
mm3d Nuage2Ply Result2/NuageImProf_STD-MALT_Etape_6.xml Attr=21.jpg RatioAttrCarte=4

mm3d Malt GeomImage "(10|2[3-4]).jpg" Abs1 Master=10.jpg DirMEC=Result4 UseTA=1 ZoomF=4 ZoomI=32 \
      Purge=true DefCor=0.001
mm3d Nuage2Ply Result4/NuageImProf_STD-MALT_Etape_6.xml Attr=10.jpg RatioAttrCarte=4
```

The first three lines are now well known : search for homologous points in all pictures to work in a common framework (`Tapioca`), followed by the camera calibration and position identification (`Tapas`). The `position_UTM33_cut_Zcst.csv` ASCII file providing the position data as columns is not appropriate for MicMac which is expecting an XML formatted files : the conversion is performed by `OriConvert` which is used to generate a new orientation directory including the camera positions named `Ori-jmfgps`. The pictures processed in an arbitrary coordinate system (`Init1`, resulting from the last call to `Tapas`) are converted to the absolute projected GPS coordinate framework (`jmfgps`) in order to generate the new orientation directory `Abs0`. GPS positions are however flawed, and the camera positions computed by MicMac (in its arbitrary framework) don't exactly fit to the noisy GPS positions. Weighting both contributions – photogrammetric and GPS position measurements – is performed by `Campari` which uses as input the absolute orientation directory `Abs0` in order to generate the new framework `Abs1` used for computing the final dense point cloud by `Malt`. We had to split the picture dataset into three distinct subsets in order to prevent MicMac from mixing features found on different walls and inconsistently matching homologous points.

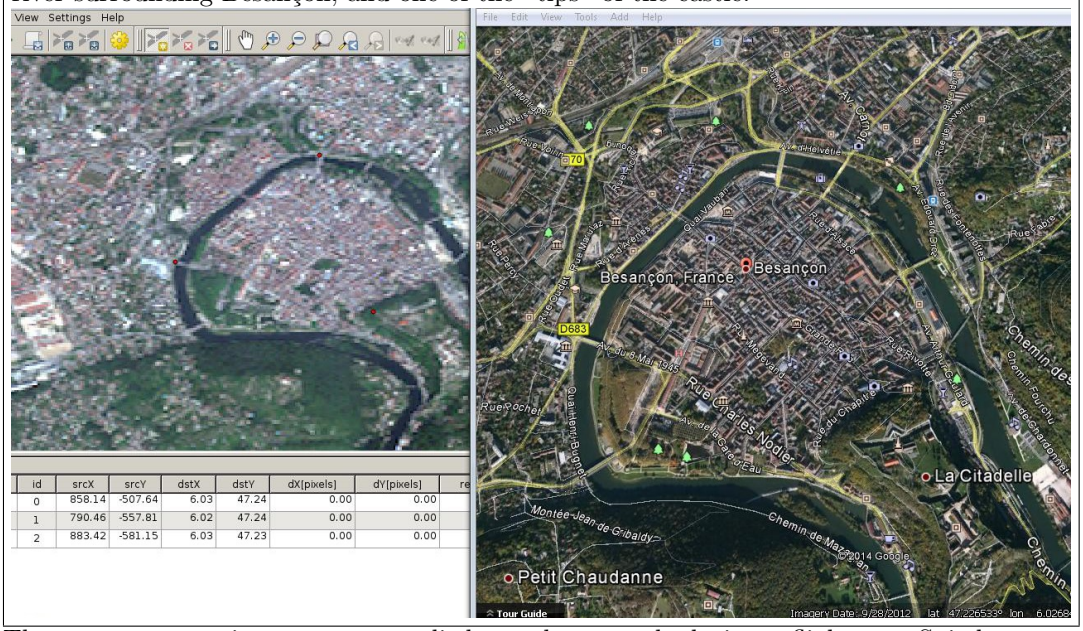
The three point clouds are inserted and merged in CloudCompare, which was selected over Meshlab since the latter tool does not keep the color information when exporting the global point cloud to ASCII format (file with XYZ extension). A few lines of the resulting file are

```
508.69662476 255.29411316 316.57540894 253 254 255
508.73446655 255.26551819 316.57797241 252 254 254
508.70834351 255.15536499 316.51879883 232 237 241
508.74603271 255.12689209 316.52133179 192 202 212
```

which seem reasonable since X and Y are respectively in the 500 and 250 m range respectively, consistent with the camera positions we have provided. We add to these first two columns the offset values of 728000 and 5235000 respectively to return to the absolute framework, load this dataset in QGis (`Layer` → `Add Delimited Text Layer`) over a satellite image background satellite (`https://browse.digitalglobe.com`, color image from GeoEye1 over Besançon dated August 7th, 2009 and quickly georeferenced by using the coordinates of three bridges acting as reference points selected in Google Earth) and a digital elevation model (SRTM) for validating the proper positioning and orientation of the resulting point cloud (Fig. 24). Although the GeoEye1 satellite is claimed to provide a 1.84 m on the final color product, the degraded preview image available on the Digitalglobe web site exhibits, in the Besançon area, a pixel size estimated to be around 11 m×17 m – or more or less the same resolution as the Landsat7 images (`http://landsat.gsfc.nasa.gov/?page_id=2376` and `http://landsatlook.usgs.gov/` for the links towards the two agencies responsible for this program – NASA and USGS). A video summarizing this operation sequence aimed at loading the point cloud in QGis and coloring the dataset as a function of the altitude of each point is provided at `http://jmfriedt.free.fr/micmac_qgis.mp4`. In this example, the height attribute defines the color of each point by selecting in the properties of the vector layer the `Graduated` symbol (instead of `Single Symbol`) and using the Z `Column` with as many classes as wanted. If an the other hand we wish to keep the original color of each point, we keep the `Single Symbol`, and, after clicking on `Simple Marker`, selecting `Data defined properties` to enter the formula defining the `Fill Color` with `color_rgb(R, G, B)`.

Reminder about satellite image georeferencing

A detailed description of the procedure for georeferencing satellite images based on some known ground control point positions and rubber sheeting of the image was provided earlier [1]. This operation is performed by the `georeferencer` plugin of QGis by exploiting GPS positions acquired from Google Earth on a few GCPs, in this case the bridges over the Doubs river surrounding Besançon, and one of the "tips" of the castle.



These same processing steps are applied on a larger scale during a flight over Spitsbergen, allowing for a quantitative analysis of the resulting model (Fig. 25), excellent in the abscissa and ordinate plane. However the result is, in this example, very poor in the altitude direction due to an average plane

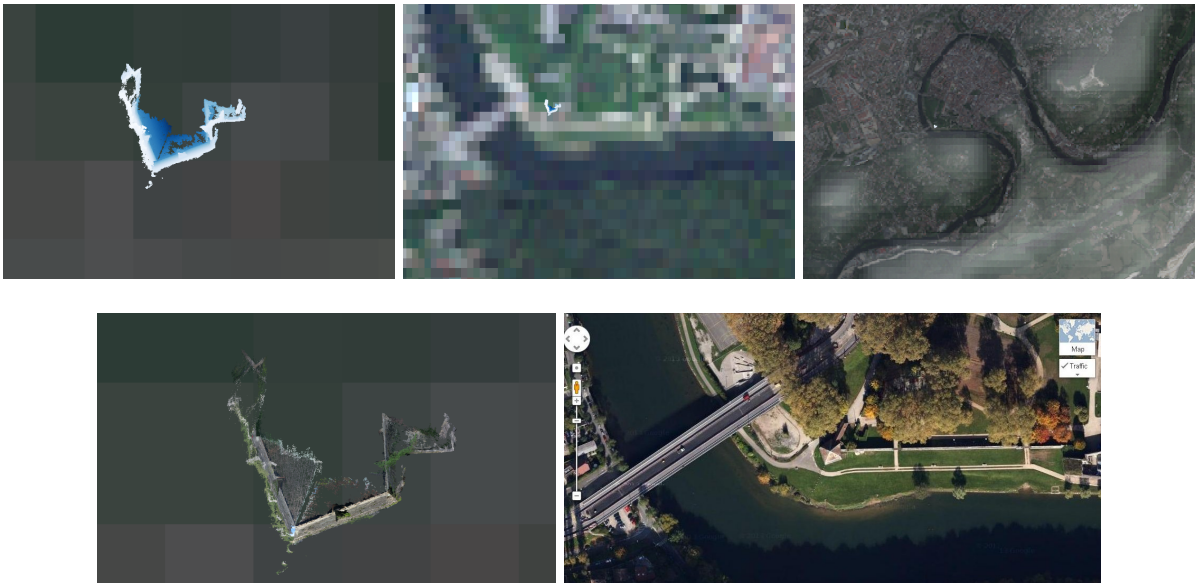


FIGURE 24 – Inserting the point cloud of the tower in QGis to provide the geographical context – here a GeoEye1 satellite image and a digital elevation model visible as a partly transparent layer. The tower is properly located near the Charles de Gaulle bridge (top, middle), as confirmed by the Google Maps screenshot (bottom, right). QGis allows coloring each point with the RGB fields saved by CloudCompare in the point cloud description, hence restoring the original color of each pixel (bottom, left).

whose altitude is not constant. The tilted plane is the result of the linear flight path of the plane as these pictures were shot. Other examples in which the plane is turning over the target yield much better results in the elevation direction, despite large altitude errors of the GPS coordinates. We have not aimed at subtracting this average plane due to the lack of robust ground control points and a poorly defined coast line on these pictures. This limitation is best assessed *before* planing a flight for acquiring aerial images, a costly operation best not missed unless picture acquisition requirements are not properly understood.

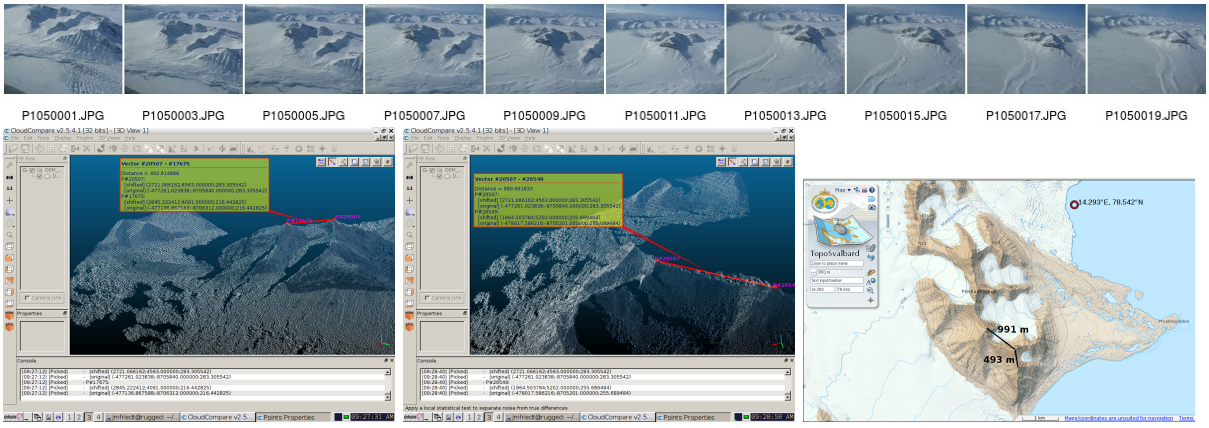


FIGURE 25 – Top : picture set for computing a digital elevation model. Bottom, left and middle : measurement of the distance between two summits after processing the point cloud, including the GPS coordinates of the camera along the flight path. Bottom, right : measurement of the distance between the same summits on the reference map `toposvalbard.npolar.no`. The red dot indicates one of the plane positions while shooting these pictures.

9.3 Resolution

The reader should be aware of possible computation errors if the Cartesian coordinate origin is not brought close to 0. In the example below in which pictures of one of the buildings of the Besançon university (Fig. 26) next to the Doubs river were shot with OSMTracker running on a Samsung S3 mobile phone, the pictures are located around (47.23°N, 6.02°E, 289 m) or, in the UTM31N framework (728357 m, 5235782 m, 289 m). Computing the dense point cloud in the absolute framework yields a catastrophic result due to rounding errors, even when using floating point numbers. Bringing the framework origin close to 0 (by subtracting to 728000 abscissa and 5235000 to ordinates) yields a correct resulted properly graduated in meters. This point cloud is then brought back to its proper position in the absolute WGS84/UTM31N framework by adding back the offset to the coordinates : the XYZ format of the point cloud saved by CloudCompare also includes three color columns (RGB) and is read by GNU/Octave in order to perform arithmetic operations on the first two columns.

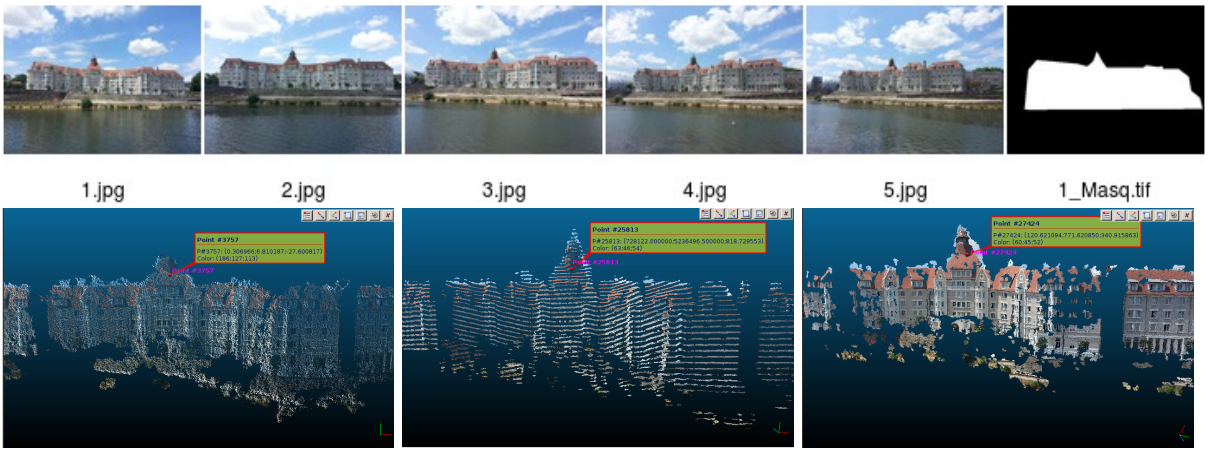


FIGURE 26 – Top : picture set of a university building in Besançon, with the mask used to restrict the computation to the building wall and exclude the clouds in the sky and the river as the rightmost image. Bottom : result of the computation in an arbitrary framework in which the result is excellent but dimensions are not usable. Middle : the coordinate system is graduated in meters but the point cloud resolution is poor due to computation rounding errors. Right : absolute framework and excellent resolution are achieved by bringing the origin close to 0. In all cases, the bubble includes the coordinates of one of the points in the tower : notice how the displayed coordinate evolves from the arbitrary framework (abscissa/ordinate below 10 and negative altitude) to an absolute framework (altitude around 800 m, erroneous due to computation errors) and finally the absolute framework brought close to the origin.

10 Data dissemination

Tools for displaying point clouds discussed so far must be specifically installed on the user computer, hence requiring a voluntary action of downloading the point cloud file and loading it in the dedicated software. Offering the option of viewing the point cloud through a web interface might help promote the dataset quality and induce downloading the dataset for local handling. The time when VRML was popular for displaying vector data on web browsers seems over, and it seems that the WebGL protocol included in HTML5 now provides such functionalities. Let us emphasize from the beginning that the next paragraph will very probably yield web browser crashes : bookmarks should be safely stored before accessing the web pages cited below.

A first limitation of WebGL, which appears to be related to a compatibility issue with mobile platforms such as mobile phones and tablets, is the maximum number of points displayed : 64 Ksamples. Hence, as was the case for GPS tracks in KML format to be included in Google Earth, each dataset will have to be split in many subsets complying with this size requirement. A free tool for dis-

playing point clouds simple enough for our purpose is `potree`[18]. This set of JavaScript programs is executed by a web server to display a point cloud through an user friendly interface, assuming the point cloud was converted to the appropriate format, complying with WebGL limitations : the `PotreeConverter`[19] tool is provided for this purpose. Our tests were limited to cloud points saved in the ASCII formatted 6-column file storing data as XYZRGB (*very* large temporary file) since all attempts to convert PLY files (binary or ASCII) have failed, resulting in a segmentation fault error message. After saving the point cloud resulting of merging all point clouds (multiple master images) as an an ASCII file with `Cloudcompare`, we generate a directory including all datasets by running `potree` with the following command line : `PotreeConverter nuage.xyz -f xyzrgb -r 255 -s 0.1 -l 4` where `-s` must be tuned to match the requirement of each point cloud (would be rather of the order of unity for point clouds georeferenced on the GPS framework, here the point cloud extends on a ± 10 range in all three directions with 8 decimals). An HTML file including the JavaScript calls to the `potree` libraries includes the position of the camera (`camera.position.x` etc ...) and the path for loading the point cloud (`pointcloudPath="../resources/pointclouds/granvelle2/cloud.js"; POCLoader.load(pointcloudPath);`). The more than 50 millions points included in the cloud, representing the Granvelle Palace in Besançon (build in the 16th century, and now hosting the Time Museum), only requires a disk space of 50 MB on the web server.

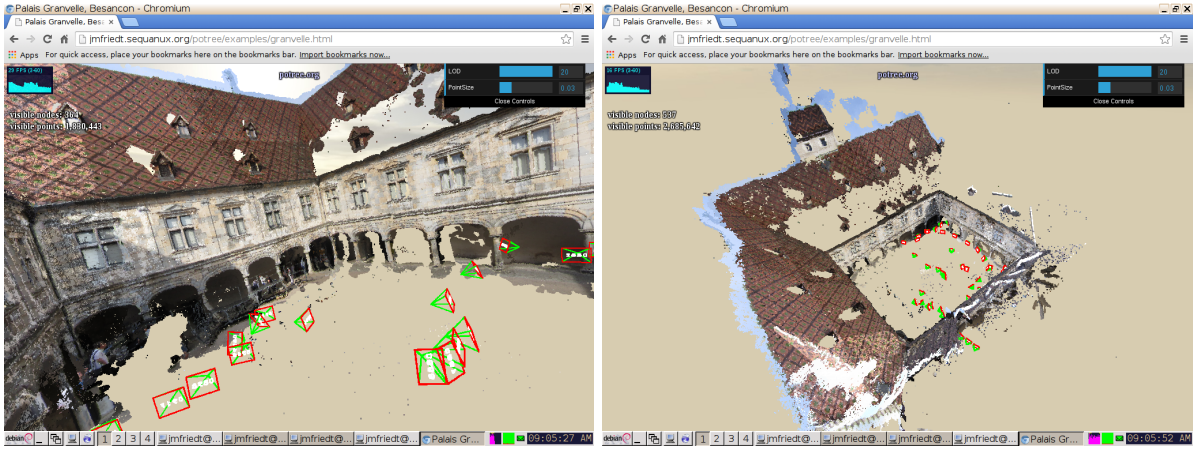


FIGURE 27 – Display of a point cloud of over 4 million points by using the `potree` scripts executed by the web server hosted at `jmfriedt.sequanux.org/potree/examples/granvelle1.html`, here loaded by a `chromium` web browser, one of the few software not crashing when accessing such a dataset.

The last missing step at the end of this discussion is related to our initial aim of converting virtual point clouds to real objects by using 3D printing techniques, a fashionable topic at the moment in amateur circles but well established in professional fields (the entrance of ENSG (École Nationale des Sciences Géographiques) exhibits a beautiful 3D model of the Chamonix valley next to the Mont Blanc summit) : the conversion of the point cloud to a surface ready for printing appears more challenging than expected.

11 Conclusion

We have described the use of a free software provided by IGN – MicMac/Apero – for processing digital pictures of objects observed from various points of view in order to generate three-dimension point clouds modeling the observed scene. This tool, initially designed for mapping building walls or for generating digital elevation models from aerial or oblique view pictures, includes all the necessary steps to generate a quantitative mode – either by including well known ground control points, or the position of the camera when each picture was shot.

18. `https://github.com/potree/potree`
19. `https://github.com/potree/PotreeConverter`

We have identified some of the conditions to be met for maximizing chances of successfully generating a point cloud our of the acquired pictures : as with all digital signal processing, a poor input data source will be unable to generate a usable result. However, in order to allow most of the audience to become familiar with this tool, we have aimed at only using image acquisition systems widely available such as mobile phones, compact digital camera, or a webcam flying on a drone. Obviously, results would be expected to be greatly improved by using a reflex camera with a state of the art lens – especially the correlation maps would become uniformly white and display far fewer dark areas than on our examples.

Finally, in order to reach beyond the qualitative result without loosing the fun part of the topic, we have including the resulting point clouds in a Geographical Information System (GIS) by adding the newly generated datasets to existing digital elevation models or freely available satellite images (with a resolution much poorer than those provided by our point clouds when observing buildings).

Perspectives for using these tools are nearly infinite and reach far beyond geography related fields, being applicable to engineering or 3D reconstruction of mechanical pieces (or, in our case, toys). Extending to more exotic image acquisition systems – for example scanning electron microscopes – which no longer comply with the assumptions implemented in `Tapas` and `Tapioca`, requires a detailed understanding of the software and the underlying theory, which is beyond the capability of this author. However, the availability of the source codes keep the possibility of extending the functionalities of MicMac open : the reader is invited to become familiar with the basics and adapt to his/her own needs.

Acknowledgements

J.-M Friedt is an engineer in a private company, hosted by the Time and Frequency department of the FEMTO-ST institute in Besançon (France), a partner to the French National Research Agency (ANR) CryoSensors project, who funded the participation to the excellent MicMac tutorial taught at l'ENSG, as well as the trip to Norway during which datasets exhibited in Figs. 2 et 23 were acquired. This tutorial was written before the teaching at ENSG which provided the theoretical and practical basics to correct some of the issues that had only been presented superficially in the initial draft of this manuscript. We have however aimed at not including any newly learnt topic which had not been envisioned prior to the course by reading documents freely available on the web.

My geograph colleagues – D. Laffly (GEODE, Toulouse), F. Tolle & É. Bernard (ThéMA, Besançon) have informed me of the existence of MicMac/Apero. S. Gascoin (CESBIO/CNES, Toulouse) provided a significant amount of the literature references. J.-P. Simonnet (Laboratoire de ChronoEnvironnement, Besançon) provided the aerial pictures acquired from an ultralight motorized plane for generating Figs. 8 to 10. Matthieu Deveau (IGN) has patiently answered to my questions. É. Carry (FEMTO-ST, Besançon, president of the Sequanux association for the promition of free software) installed the `potree` Javascript software on the `sequanux.org` web server.

The web site `gen.lib.rus.ec` provided by literature references not freely found on the web : all my amateur (or professional) research would be impossible without this database of scientific and technical documents. All references related to the author are available at `http://jmfriedt.free.fr`.

Références

[1] J.-M Friedt, *Correction géométrique d'images prises en vue oblique – projection sur modèle numérique d'élévation pour exploitation quantitative de photographies numériques*, GNU/Linux Magazine France n.167, Janvier 2014, pp.42-57

[2] F. Remondino, S. del Pizzo, T. Kersten, S. Troisi, *Low-cost and open-source solutions for automated image orientation – a critical overview*. In : M. Ioannides *& al* (Eds.), *Progress in Cultural Heritage Preservation, Lecture Notes in Computer Science*, **7616**, Springer, Berlin Heidelberg, pp. 40–54 (2012)

[3] voir par exemple l'intitulé de la session "Cryospheric applications of modern digital photogrammetry from airplane, UAV, and ground-based instrument platforms" de l'American Geophysical

Union à `https://agu.confex.com/agu/fm14/webprogrampreliminary/Session3014.html`, et plus généralement le blog de Matt Nolan à `http://www.drmattnolan.org/photography/2014/`

[4] J.-M Friedt, *Auto et intercorrélation, recherche de ressemblance dans les signaux : application à l'identification d'images floutées*, GNU/Linux Magazine France **139** (Juin 2011)

[5] D.G. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision, **60** (2) pp.91-110 (2004)

[6] Y. Egels & M. Kasser, *Digital Photogrammetry*, CRC Press (2001), et en particulier le chapitre 3 de cet ouvrage qui décrit les méthodes de réduction de l'espace des recherches de points homologues compte tenu des paramètres géométriques des caméras

[7] la documentation de MicMac se trouve dans le répertoire `Documentation` de l'archive mercurial (fichier `DocMicMac.tex`)

[8] W.W. Immerzeel, P.D.A. Kraaijenbrink, J.M. Shea, A.B. Shrestha, F. Pellicciotti, M.F.P. Bierkens, S.M. de Jong, *High-resolution monitoring of Himalayan glacier dynamics using unmanned aerial vehicles*, Remote Sensing of Environment 150 (2014) 93–103

[9] M.J. Westoby, J. Brasington, N.F. Glasser, M.J. Hambrey, J.M. Reynolds, *"Structure-from-Motion" photogrammetry : A low-cost, effective tool for geoscience applications*, Geomorphology 179 (2012) 300–314

[10] I. Colomina, P. Molina, *Unmanned aerial systems for photogrammetry and remote sensing : A review*, ISPRS Journal of Photogrammetry and Remote Sensing 92 (2014) 79–97

[11] J.-M. Friedt, É. Bernard, F. Tolle, D. Laffly, *Gestion d'Informations Spatialisées : outils libres pour l'exploitation de données géolocalisées – au-delà des aspects géographiques*, séminaire CETSIS 2014 (Besançon, France), available at `http://jmfriedt.free.fr/cetsis_sig.pdf`

[12] J.-M. Friedt, *Géolocalistion de photographies numériques*, GNU/Linux Magazine France 96, Juillet/Août 2007 – noter la mise à jour du script de géolocalisation des photographies numériques suite au passage à la version 3 de l'API de Google Maps – bien que la méthode décrite dans l'article reste la même, l'implémentation est mise à jour dans `http://jmfriedt.free.fr/photos_asc.txt`. Par exemple, `http://jmfriedt.sequanux.org/130929_kvad/photos_asc.php`.

[13] J. Lefort, *L'aventure cartographique*, Belin–Pour la Science (2004)