Terrain 3d reconstruction drone using SfM

Ali Yaghi[1]

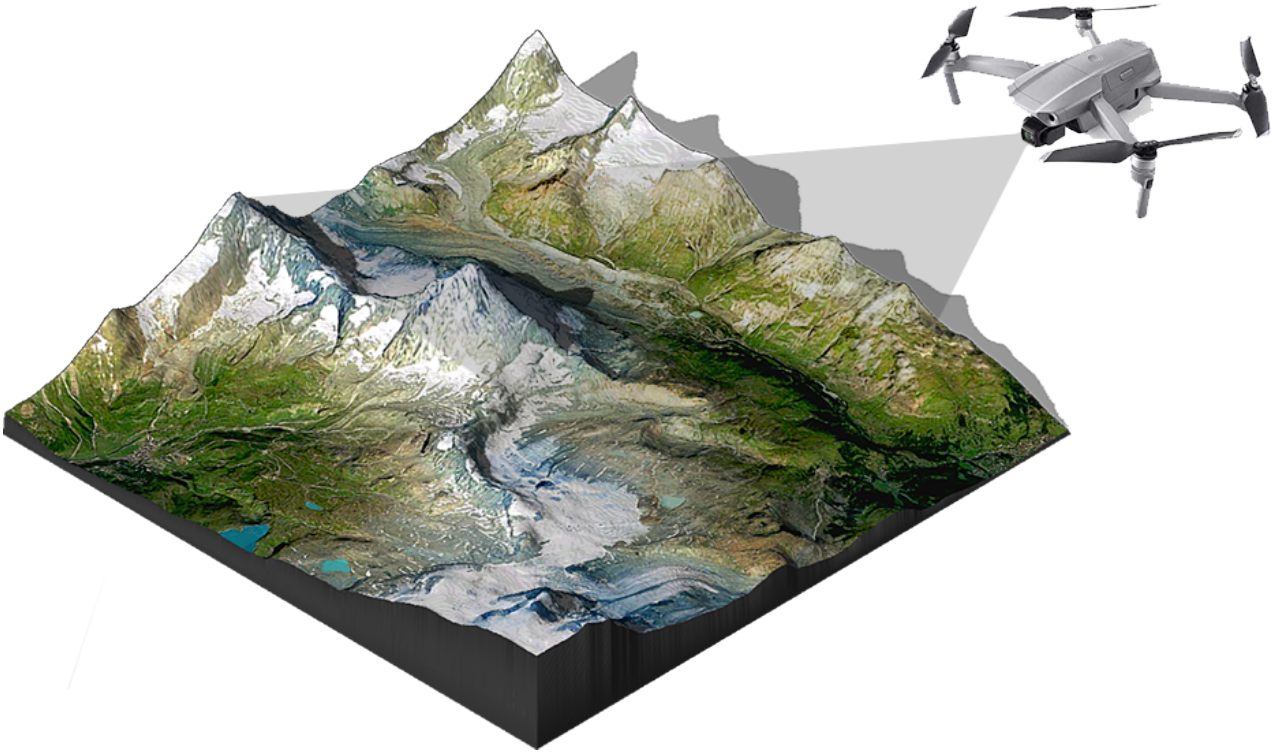
[1]**91000 Evry, France | ali.yagi99@gmail.com**
[1]**Université Paris-Saclay, Université d'Evry Val d'Essonne**

Abstract

3d reconstruction was introduced to infer the 3D geometry and structure of objects and scenes from one or multiple 2D images. Structure from motion using multi-view stereo, a photogrammetry and computer vision technology that employs overlapping photos to reconstruct 3D surface models, is a useful research tool in geomorphology and related fields. we propose in this paper the SfM technique to be able to 3d reconstruct terrains and recognize its Milestones.

Keywords

Structure from motion | 3d reconstruction | drones | terrain | camera calibration



1. Introduction

Photos-based 3D reconstruction attempts to deduce the 3D geometry and structure of objects and scenes from a single or many 2D image. Many applications rely on this long-standing ill-posed problem, including robot navigation, object identification and scene interpretation, 3D modeling and animation, industrial control, and medical diagnostics. There are several strategies for creating a 3D model of a landscape; however, all of them are either expensive or inaccurate due to the usage of sensors and high-quality cameras.

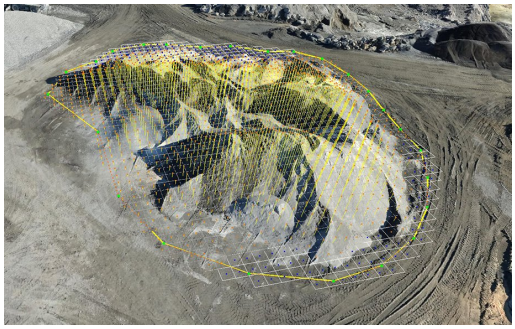


Figure.1 Terrain estimation

Structure from motion using multi-view stereo, a photogrammetry and computer vision technology that employs overlapping photos to create 3D surface models, is a useful research tool in geomorphology and related fields. SfM is a low-cost technique that complements other 3D technologies like terrestrial and airborne laser scanning since images may be gathered using common consumer-grade cameras (lidar). The high level of automation in SfM processing provides a once-in-a-lifetime opportunity to explain earth surface processes, but it also comes with advantages and disadvantages. As a result, the goal of this contribution is to provide a roadmap for effectively applying SfM to a variety of geomorphic research. It begins by providing an overview of the approach, including its history, evolution, and the reasons for its success. Second, it explains the approach, including recommendations for appropriate settings, accuracy, and georeferencing. Finally, it highlights the opportunities for reconstructing processes across spatial and temporal dimensions by providing case examples given by specialists from around the world.

2. Structure from motion

There are 4 main steps for 3d reconstruction using SfM

2.1 Camera Calibration

As we mentioned earlier, SfM uses one camera to capture a video, this camera should be calibrated, in other word we should know it's intrinsic and extrinsic parameters in order to have better result. Camera calibration is often introduced due to lack of accuracy in manufacturing, this usually happens for cheap cameras.

To begin, we define a camera as an electrical instrument that digitally records an image. As a real camera projection, the pinhole camera concept is commonly utilized. This model is based on the collinearity principle, which states that each point in object space is projected onto the picture plane by a straight line via the projection center. It's a valuable model since it allows for a straightforward mathematical expression of the link between object and picture coordinates. When great precision is desired, however, it is not valid, and a more thorough camera model must be utilized. Extrinsic and intrinsic parameters are separated into the model's parameters. Camera posture is referred to as extrinsic parameters, whereas picture plane features are referred to as intrinsic parameters. A best way to calibrate a camera is to capture images for a chessboard of known dimensions and try to guess the parameters.
The following two images show a cheap camera captures before and after calibration.

Figure.2 Chessboard correction

The equations that relate 3D point (X_w, Y_w, Z_w) in world coordinates to its projection (u, v) in the image coordinates are shown below.

$$\begin{bmatrix} u' \\ v' \\ z' \end{bmatrix} = \mathbf{P} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \qquad u = \frac{u'}{w'} \qquad v = \frac{v'}{w'}$$

Where, **P** is a 3×4 Projection matrix

$$\mathbf{P} = \overbrace{\mathbf{K}}^{\text{Intrinsic Matrix}} \times \overbrace{[\mathbf{R} \mid \mathbf{t}]}^{\text{Extrinsic Matrix}} \qquad \mathbf{K} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

With regard to the object coordinate system, the camera coordinate system's origin is at the projection center at {x0, y0, z0} and the camera frame's z-axis is perpendicular to the picture plane. The rotation is represented using Euler angles that define a sequence of three elementary rotations around x, y, z-axis respectively. we do the rotations clockwise, first around the x-axis, then the y-axis which is already once rotated, and then around the z-axis, which has been rotated twice in the preceding steps. To represent an arbitrary object point P in image coordinates at Xi, Yi, Zi, we must first translate it to camera coordinates {xi, yi, zi}.
The goal of the calibration process is to find the 3×3 matrix **K**, the 3×3 rotation matrix **R**, and the 3×1 translation vector **t** using a set of known 3D points (X_w, Y_w, Z_w) and their corresponding image coordinates (u, v).

2.2 Features detection

SfM works on multiple images, we should try to guess the correspondence between images to register it and have a well reconstructed model, but first we should detect features of each image separately. There are many methods for detecting features such as ORB, SIFT, SURF, AKAZE, BRIEF, FAST, HARRIS CORNER DETECTION, etc. We used in this paper the well-known SIFT method to detect and describe local features (keypoints) in images (Lowe, 2004).

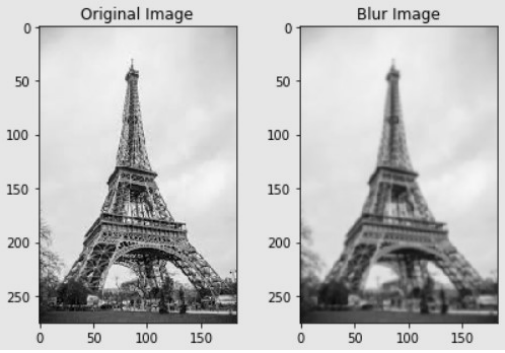
SIFT (Scale Invariant Feature Transform), is a feature detection algorithm in Computer Vision, it helps locate the local features in an image, commonly known as the 'keypoints'(Kp) of the image. These **Kp**s are invariant to scale & rotation and can be used for various computer vision applications, like image matching, object detection, scene detection, etc.

We can use also **Kp** generated by SIFT as features for the image during model training. The major advantage of SIFT features, over edge features or hog features, is that they are not affected by the size or orientation of the image.

We must find the most distinguishing features in a picture while ignoring any noise. We also need to make certain that the features aren't scale-dependent. These are important topics, so we will talk a little bit about it.

to reduce noise in an image, we will be using Gaussian Blur technique.

So, for every image's pixel, the Gaussian Blur calculates a value based on neighbor pixels. Below is an example of image of Eiffel tower before and after the Gaussian Blur. As we can see, the texture and minor details are removed and only the relevant information like the shape and edges remain in the image:



We were able to eliminate the noise from the photos using Gaussian Blur, and we were able to emphasize the main parts of the image. Now we must confirm that these characteristics are not scale-dependent. By building a 'scale space,' we'll be able to search for these properties on numerous sizes.

Scale space is a collection of photos that have different scales, generated from a single image.

As a result, these blur pictures are formed in a variety of scales. We'll take the original image and lower the scale by half to make a new series of photos with varying scales. As we saw above, we'll make blur versions of each new image.

So far, we've constructed photos with several scales (typically denoted by σ) and utilized Gaussian blur to minimize noise in each of them. Then, using a technique known as Difference of Gaussians (DoG), we'll try to improve the features.

Difference of Gaussian is an algorithm to enhance features that makes the subtraction of one blurred version of an original image from another, less blurred version of the original.

DoG creates a new set of images, for each octave, by subtracting each image from the previous one in the same scale.

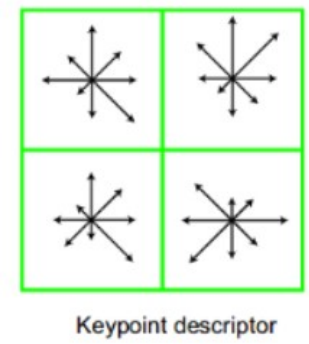
once we create the images, the following step is to identify the image's main **Kp**s that may be utilized for feature matching. The goal is to locate the pictures' local maxima and minima. Find the local maxima and minima and delete low contrast **Kp**s are the two stages in this section (**Kp** selection).

To deal with low contrast **Kp**s, a second-order Taylor expansion is computed for each **Kp**. If the result is less than 0.03 (in magnitude), we reject the **Kp**.

We will now assign an orientation to each of these **Kp**s to make it invariant to rotation. This can be done by calculating magnitude and orientation and creating a histogram for it:

$$Magnitude = \sqrt{Gx^2 + Gy^2}$$
$$\phi = \arctan(Gy/Gx)$$

Now for the last step of SIFT, we have stable **Kp**s that are scale-invariant and rotation invariant. we will use the neighbor pixels, their orientations, and magnitude, to generate a unique fingerprint for this **Kp** called a 'descriptor'.



here is an example of SIFT on an image of Université d'Evry.

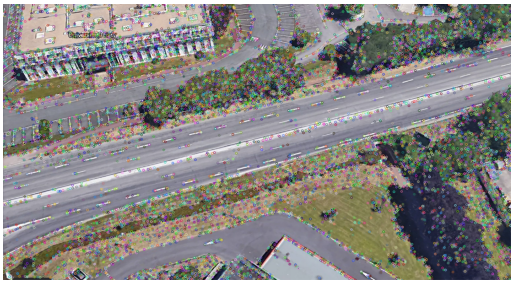


Figure.3 université d'Evry

2.3 Images matching

Since we should find the same point on many images to be able to estimate the pose of this point in the 3D world, we do match features that are extracted in the 2nd step. Each **Kp** is described by a vector 128- dimensional. Using the "Approximate Nearest Neighbors" package of Arya, et al. (1998), we applied the SIFT approach to generate an automated feature matching over all picture pairs: - In the space of 128-dimensional vectors, we use Euclidean distance. - For effective near neighbor search, we employ kd-trees. - We compare the distance to the database's best-matching SIFT feature and the distance to the second-best-matching feature.. If the ratio of closest distance to 2nd closest distance is greater than 0.8 then reject as a false match.

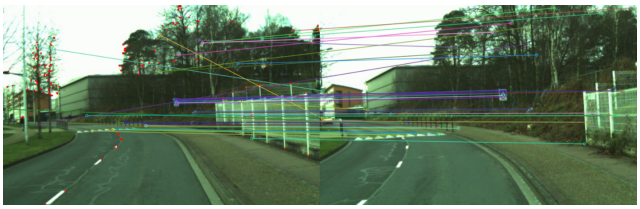


Figure.4 features matching using RANSAC

2.4 Bundle Adjustment

Bundle adjustment guarantees that observations of a single ground feature in successive photos are consistent. If they aren't, then the cameras' location and orientation, as well as the feature's 3D position, must be modified until they are. This optimization is carried out in conjunction with thousands (if not millions) of other comparable restrictions employing a wide range of characteristics seen in other photos. Bundle adjustment is extremely strong and adaptable: it may work with as little as two overlapping photos or as many as thousands. It's also a potentially harmful instrument. To ensure and verify that the answer accurately represents reality, careful attention is essential.

Recent research in Structure from Motion has shown that large-scale community picture sets may be used to rebuild geometry. Bundle adjustment, or the non-linear joint refinement of camera and point parameters, is a critical component of most SfM systems, and it can take a long time for big tasks. The scalability of bundle adjustment algorithms has become a major issue as the number of photographs in such collections continues to expand into the hundreds of thousands or even millions.

After all of the frames (pictures) have been added and matched, we construct a regularization approach for the final camera postures using Lourakis and Argyros' sparse Levenberg-Marquardt algorithm (2004). We repeat this process until our error (back-projection error) converges. The Bundle Adjustment technique refines the original 2D-to-3D correspondence, which is then utilized to compute the final camera posture.

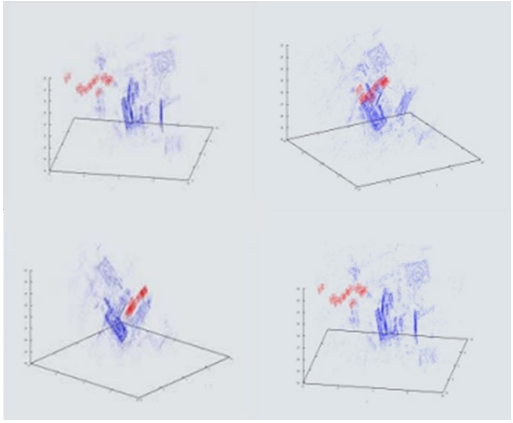
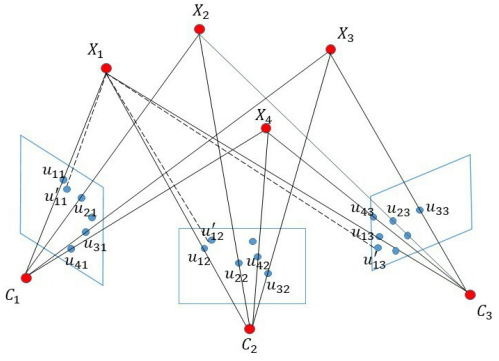


Figure.5 bundle adjustment progress

The 3D point-cloud scene reconstruction is carried out, using both 2D camera views and 3D camera-specific info. The task of improving a visual reconstruction to provide simultaneously optimum 3D structure and viewing parameter (camera posture and/or calibration) estimations is known as bundle adjustment. The parameter estimates are determined by minimizing some cost function that quantifies the model fitting error, and the solution is optimum in terms of both structure and camera variations.
Bundle adjustment entails fine-tuning a set of initial camera and structural parameter estimations in order to discover the set of parameters that best forecast the positions of observed points in the collection of accessible pictures. More formally, let's assume that n 3D points are seen in m views and let X_{ij} be the projection of ith point on the image j. Let v_{ij} denote a binary variables that are equal 1 if the point i is visible in an image j and 0 otherwise. Assume that each camera j is parameterized by a vector $\mathbf{a}_j$ and each 3D point i by a vector $\mathbf{b}_i$. Bundle adjustment will minimize the total reprojection error with respect to all 3D point and camera parameters, specifically

$$min_{a_j,b_i} \sum_{i=1}^{n} \sum_{j=1}^{m} v_{ij} d(Q(a_j, b_i), x_{ij})^2$$

where $Q(a_j, b_i)$ is our predicted projection of point i on image j & $d(\mathbf{x}, \mathbf{y})$ is the Euclidean distance between the image points represented by the vectors $\mathbf{x}$ and $\mathbf{y}$. Bundle adjustment is by definition tolerant of missing picture projections since the minimum is computed over many locations and many images, and if we choose the distance metric for a reason (e.g., Euclidean distance), bundle adjustment will also minimize a physically meaningful criterion.



3. Terrain Reconstruction

As we mentioned earlier, our goal is to make a drone capable to 3d reconstruct a terrain using it's camera by capturing a video and extract interesting and useful features from this video.

3.1 SfM from 2 views

to make sure that SfM will work well with us, we tried the 2 views-based algorithm using matlab, this algorithm is open and free to test, you can reach it from this link. we did some change to this code in order to 3d reconstruct a specific scene and support it's camera (camera's parameters is the most important here). However we will share some results of how good is SfM.

Figure.6 Original Image

This is our original image, we want to distort it using the matrix we have collected previously using camera calibration.

Figure.7 Distorted image

From this image, we have found 150 strong corners, we do match those corners with another image taken for the same scene but from different angle. The epipolar lines of the second image collected are shown in the next image.

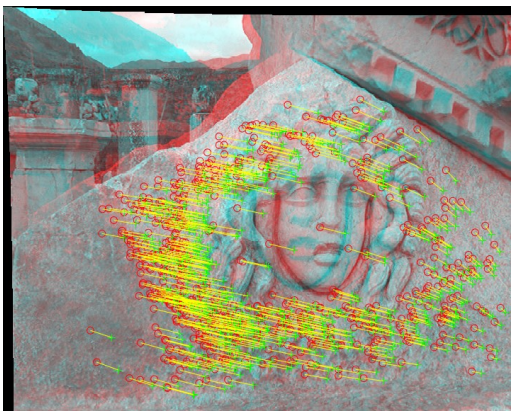


Figure.8 Epipolar lines

Finally, using what we got, we are able to estimate the pose of the camera and reconstruct a good 3d scene from the 2 captured images.

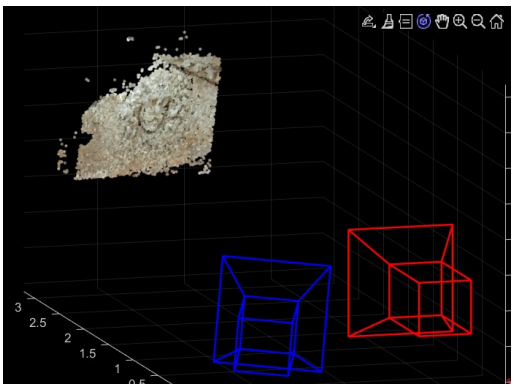


Figure.9 Three dimensional scene

3.2 SfM from multi-view

since we got a good result from only 2 images, we decided to move to SfM from multi-view, which will be helpful later when we capture a video using drone's camera. to do so, we should calibrate our camera using the method that we talked about using a known dimension chessboard, we found a 25mm chessboard, took 9 photos from different angles and tried to detect its corner. It seems obvious that everything is okay.

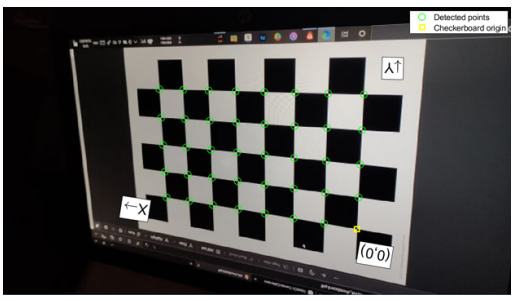


Figure.10 25mm chessboard

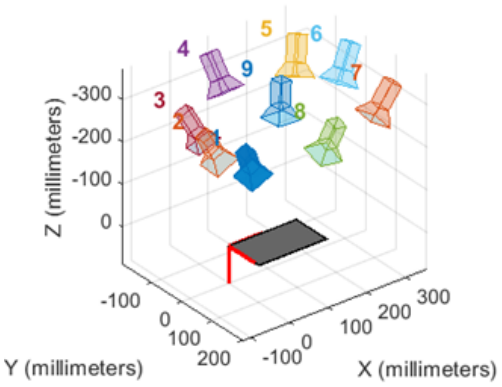


Figure.11 Camera's positions

The camera's positions 3d representation shows an accurate and successful calibration for our camera, we also got the important parameters that we can't proceed without.

517.6182	0	0
0	517.1610	0
342.9652	192.8342	1

After that, we detect features, match each pair of images using those features, and since not all matches are useful we neglect some outliers to enhance our code, those outliers are helpless and can slow down the process of 3d reconstruction.
Matches list is Three dimensional, each phase contains the row and columns of matching between each pair of image, hence, this leads us to make a nested loop O(n3), but it seems okay since the computation does not take a lot of time if images are not in high resolution. Some results of what we did are show below before and after outliers removal.

```
num_matches before outlier removal: 59518
Number of img pairs is 1035 out of possible 1035
After outlier removal: 47762
Number of img pairs is 245 out of possible 1035
```

Last movements was about first Three step in structure from motion, we now have clean matching so we can do the last and most important step which is bundle adjustment.
In bundle adjustment we should keep an eye open for the next pair of images to be treated, each pair will be used to grow our reconstructed model but the next pair should be known to continue the growing after the old one, this can help us to refine and build our reconstructed model better. We grouped also each pair of images into two categories, resected and unresected, images can be categorized using a function that give the best pair to work with, usually unresected images are not likely to advance with in the moment, however, after many iterations, unresected images will have it's own turn and be resected for sometime, this is because we reached the camera position that is related to those images. We now have well defined strategy to iterate on pairs and extract important 3d points, the next step is to triangulate and estimate 3d points. Triangulation consists of computing the 3D point that corresponds to the paired homologous points. For this, the projection matrix of camera are used in both positions which are available because the scene has been previously calibrated.
Using epipolar geometry's theory, we can affirm that the projection matrix complies with the formula:

$$P = K\,[R \mid t] \Longrightarrow \begin{aligned} (p_{1,1}^T - j\,p_{1,3}^T)\,\tilde{M} &= 0 \\ (p_{1,2}^T - i\,p_{1,3}^T)\,\tilde{M} &= 0 \\ (p_{2,1}^T - j'\,p_{2,3}^T)\,\tilde{M} &= 0 \\ (p_{2,2}^T - i'\,p_{2,3}^T)\,\tilde{M} &= 0 \end{aligned}$$

The second system of equations is derived using the first equation and the visual description of homologous points, which will be solved using algebraic minimization.
Some results of errors that we have obtained and seems small which give a good impression of what we did are shown here:

```
Average reprojection error on image 3 is 0.13341395860547478 pixels
Average reprojection error on image 41 is 0.1717499674398783 pixels
Average reprojection error on image 2 is 0.13199036003798348 pixels
Average reprojection error on image 42 is 0.1409799256965663 pixels
Average reprojection error on image 1 is 0.13422209420707265 pixels
Average reprojection error on image 43 is 0.13565324865156378 pixels
Average reprojection error on image 0 is 0.15110587424843383 pixels
Average reprojection error on image 44 is 0.1561807483830394 pixels
Average reprojection error on image 45 is 0.3648586187752504 pixels
Average reprojection error across all 46 resected images is 0.1617613951190237 pixels
```

Taking into account all theories and algorithms we discussed, we performed the reconstruction of the scene in 3D. The final result with the original image to compare are shown in the following images:

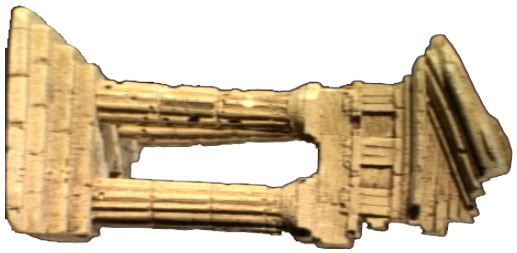


Figure.12 Original image

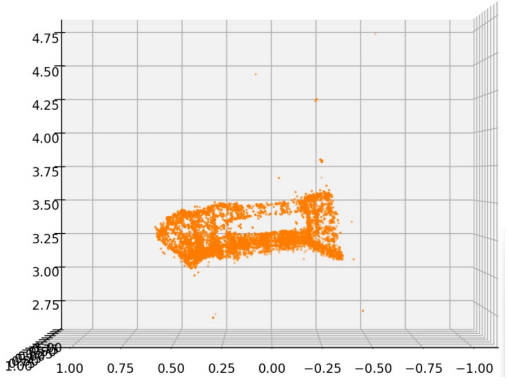


Figure.13 Reconstructed in Point Cloud format

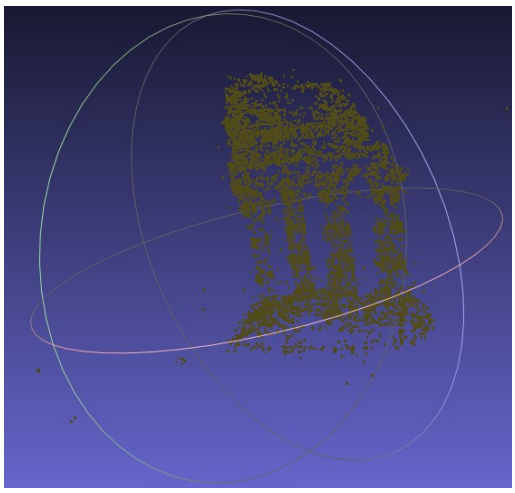


Figure.14 meshlab to better see the result

This seems to be okay, but point cloud is not preferred by eyes, so we decided to delete outliers, simplify points, create normal for point sets and meshing it.

Figure.15 meshed object

now that we succeed to have a very good result of reconstructed model, our goal is to intend it to a drone and finalizing the project.

4. Algorithm

to make it clear, here is an algorithm that explain a little bit the steps:

1.*DETECT features and match images* :

$imgs, kps, descs \leftarrow find\_features(imgset)$

$matcher \leftarrow BFMatcher(NORM\_L1)$

$matches \leftarrow find\_matches(matcher, kps, descs)$

$img\_adjacency, list\_of\_img\_pairs \leftarrow create\_img\_adj\_mat(n\_imgs, matches)$

2.*INITIALIZE Best pair, resected and unresected images.*

3.*ITERATE on images* :

while *size of unresected images* > 0 **do**

res_idx, unres_idx, prepend $\leftarrow$ *next_img_pair_to_grow_rec*

points3d_with_views $\leftarrow$ *get_correspondences_for_pnp*

Add resected images and remove unresected images.

get aligned keypoints

if $0.8 <$ *perception inliers* < 0.95 **then**

Perform Bundle Adjustment.

Triangulate images and grow the model.

if *% of inlers from Pnp is very low* **then**

Perform Bundle Adjustment again.

get and print reprojection errors.

end

4.*INITIALIZE num_voxels.*

INITIALIZE x, y, z empty.

appendx , y and z from pt3 we got from the algorithm.

add those points to a point cloud object

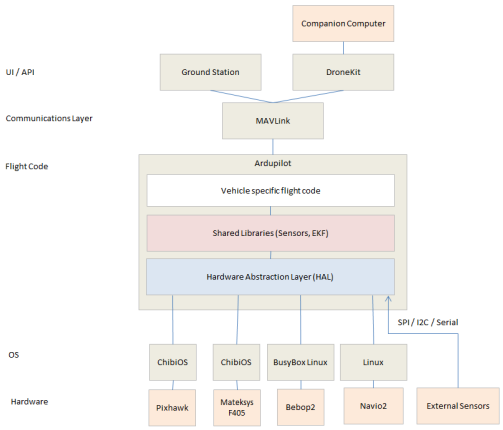
5. Drone programming

Thanks to ardupilot and many softwares that let us program drones, we are able to add the algorithm that we implemented to the drone, capture a video and process it to have the 3d model. ArduPilot is in charge of controlling a drone's hardware. Flying multirotor uavs would be impossible without ArduPilot or any other flight control firmware. That's because ArduPilot transmits the drone's motors roughly 400 orders per second, resulting in smooth and steady flying.

Figure.16 Drones

Ardupilot is essential for drone developers since it allows them to concentrate on high-level missions or applications. Let's imagine you're trying to come up with a drone delivery operation. The last thing you want to be concerned about is what PWM values are being written 400 times per second to your motors! ArduPilot takes the low-level tasks of a drone off the programmer's plate.
However, we can, rather than control the drone and capture the scene, let the drone automatically make a round over the terrain, capture the video and return to its ini-

tial point, this has been done before and succeeded with a very low error so its not a problem.
The basic structure of ArduPilot is broken up into 5 main parts: vehicle code, shared libraries, hardware abstraction layer (AP_HAL), tools directories, external support code (i.e. mavlink, dronekit).



One of the coolest aspects of the Mission Planner is that it can execute Python scripts, which is a simple method to extend the program's usefulness beyond its built-in capabilities. It can also make it simple to integrate with other dlls and modules that aren't part of the Mission Planner's initial scope. We can plan the code and start the terrain reconstruction since we developed our code in Python and used the Mission Planner capabilities.
The only problem that we face is that our algorithm is a bit heavy so we can not capture frame and process, match and use it to grow the model in real time, and drone's cpu usually is not that fast to handle all those steps before capturing next frames, so the best solution is to make it offline using post processing, we capture the video, decompose it into helpful frames and 3d reconstruct our terrain as usual.

5. Conclusions

We've devised a way for reconstructing landscape in three dimensions. This article explains how to achieve it, from feature recognition and matching through estimating geometry and improving it via bundle adjustment. 3D reconstruction is a step-by-step procedure that starts with three critical frames for initialization. Camera localization is then established for succeeding frames, and crucial frames for 3D point reconstruction are chosen. On real data, promising outcomes have been produced. The outcomes and time complexities are satisfactory. Experiments on more complicated multicamera systems and automated 3D modeling approaches employing the generic camera model are planned for the future.
It's not a surprise that many people have started to incorporate 3D technology even within their homes in the form of 3D televisions. 3D printers are becoming more widely used by the day, and some firms are even attempting to make them a normal home item. Since this technology holds great promise, it's worthy for large corporations and industries to take a look into it as well. One such industry is the construction industry.
3d drawing and modeling take a lot of time, that's what lets programmers and scientist to try to guess 3d objects using only 2d images, hence the 3d reconstruction concept created.
companies and engineers are beneficiary No.1 from 3d reconstruction, Terrain 3d reconstruction helps civil engineers to know the milestone of the campus they are working on using only 2d videos. As this technology grows and get optimized we think that the world will have a big change in term of industry.

References

[1] Gherardi, R., M. Farenzena, Fusiello, A., 2010. Improving the efficiency of hierarchical structure-and-motion. In: CVPR 2010, pp. 1594-1600, doi: 10.1109/CVPR.2010.5539782.

[2] S. Ramalingam, S. Lodha, and P. Sturm. A generic structure-from-motion framework. Computer Vision and Image Understanding, 103(3):218–228, 2006.

[3] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – A modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, Vision Algorithms: Theory and Practice, LNCS, pages 298–375. Springer Verlag, 2000.

[4] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In European Conference on Computer Vision, May 2006. 1, 2

[5] M. Brown, S. Winder, and R. Szeliski. Multi-image matching using multi-scale oriented patches. In Computer Vision and Pattern Recognition, pages 510-517, 2005.

[6] A procedure for point clouds matching from range data and image-based systems. 2017

[7] Robust bundle adjustment for large-scale structure from motion - Mingwei Cao, Shujie Li, Wei Jia, Shanglin Li, Xiaoping Liu - Computer Science - Multimedia Tools and Applications - 2017

[8] 3D Reconstruction of Indoor Scenes - F. Srajer - 2014

[9] Hamid Laga, Yulan Guo, Hedi Tabia, Robert B Fisher, Mohammed Bennamoun "3D Shape Analysis: Fundamentals, Theory, and Applications" John Wiley & Sons ISBN: 978-1-119-40519-1, 2018.

[10] Learnable Triangulation for Deep Learning-based 3D Reconstruction of Objects of Arbitrary Topology from Single RGB Images - Tarek Ben Charrada, Hedi Tabia, Aladine Chetouani, Hamid Laga - 2021