VIC PROJECT:
3D Reconstruction Using Structure-from-Motion

Mehdi Zemni
CentraleSupelec
mehdi.zemni@student-cs.fr

Chady Raach
CentraleSupelec
chady.raach@student-cs.fr

Abstract

The rising need for handling complex situations such as those witnessed in healthcare, oil and gas, games industry, mapping (Street View), and construction industries with accuracy is helping the global 3D reconstruction technology market in charting a steady curve. These technologies can also take advantage of the development of digital cameras and the increased resolution and quality of the images they produce. Even some smartphones nowadays include features that allow real time 3D reconstruction with a very good accuracy.

Structure from motion is a technique of estimating the motion of the camera and recovering three-dimensional (3D) scenes from 2-dimensional (2D) image sequences taken from two or multiple different views of one camera. The objective of this project is to identify the various approaches to generating sparse 3D reconstructions using the Structure from Motion (SfM) algorithms. First, we will describe the classical pipeline and steps of SfM. Then we will implement this algorithm step by step.

1. Introduction

In the next sections we will describe tools and techniques for obtaining information about the geometry of 3D scenes from 2D images. This task is difficult because the image formation process is generally not reversible: from its projected position in the camera image plane, a point in the scene can only be recovered up to a scale parameter corresponding to its distance from the camera. Therefore, additional information is needed to solve the reconstruction problem by exploiting, for example, points matching from different views.

In what follows, it will be convenient to work with homogenous as well as Euclidean coordinates. Formulas involving homogeneous coordinates are often simpler and more symmetric than their Cartesian counterparts. In homogenous coordinate system, points in N dimensional space are expressed in $N+1$ dimensional vector. Given a point $x = [x, y]^T$ in cartesian coordinate system, we can append a dummy coordinate to form $\widetilde{x} = [x, y, 1]^T$ in homogeneous coordinate system. To go back, we can simply devide by the last element if it is non-zero.

2. Theory

2.1. Camera Models

A camera model is responsible for transforming visual 3D world into 2D camera pixels 1 Camera models can be abstractly divided in two transformations; (1) 3D in world coordinates to 3D world in camera coordinates, and (2) 3D world in camera coordinates to 2D pixels.

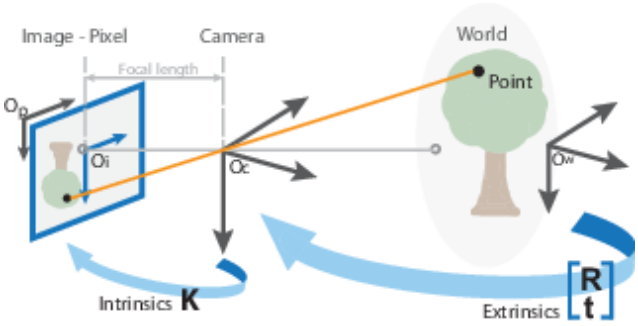

Figure 1. Camera model transformations.

2.1.1 Extrinsic Parameters: from 3D world coordinates to 3D camera coordinates

The first component of the camera model is the the rigid body transformation which transforms points from 3D world coordinates to 3D camera coordinates. This transformation preserves distances between points. Any such mapping can be represented as a composition of one translation and one rotation.

Given $\widetilde{X}_w \sim [X_w, Y_w, Z_w, 1]$ in the world coordinate system to points $\widetilde{X}_c \sim [X_c, Y_c, Z_c, 1]$ in the camera coordi-

nate system. This transformation can be expressed as:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \sim \begin{bmatrix} R & T \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (1)$$

Where R is 3x3 rotation matrix that represents camera orientation with respect to the world, and T is 3x1 vector represents the translation of camera in world coordinate system. These two parameters are called Exstrinsic Parameters.

2.1.2 Intrinsic Parameters: From 3D world coordinates to 2D pixels

Camera's intrinsic parameters can be devided into two sub-parameters:

- 3D camera coordinates to 2D image plane coordinates:
We map a point $\widetilde{X}_c \sim [X_c, Y_c, Z_c, 1]$ in 3D camera coordinates to 2D point $\widetilde{x} \sim [x, y, 1]$ on the camera image plane. The relationship between these two coordinates can be easily derived:

$$x = f\frac{X_c}{Z_c} \qquad y = f\frac{Y_c}{Z_c} \quad (2)$$

Using homogeneous coordinates we can express this in a simpler form (up to a scale):

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \quad (3)$$

- 2D image plane coordinates to 2D pixel coordinates:
To convert coordinates from image plane coordinate system ($\widetilde{x} \sim [x, y, 1]$) to pixel coordinate system ($\widetilde{u} \sim [u, v, 1]$), we use the following transformation:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (4)$$

where matrix entries are parameters intrinsic to the camera. We can combine the last two equations 3, 4 to form the camera calibration matrix K:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \underbrace{\begin{bmatrix} \alpha_u f & sf & u_0 & 0 \\ 0 & \alpha_v f & v_0 f & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{K} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \quad (5)$$

where K is an upper triangular camera calibration matrix.

Finally, it is convenient to combine equation 1 and 5 into a single formula relating 3D point $\widetilde{x}$ to its corresponding pixel position $\widetilde{u}$ through the projection matrix P

$$\widetilde{u} \sim \underbrace{K}_{\text{Intrinsic parameters}} \underbrace{[R \quad T]}_{\text{Extrinsic parameters}} \widetilde{X} \sim P\widetilde{X} \quad (6)$$

2.2. Epipolar geometry

Given 2 views of the same object, epipolar geometry shows that it is possible to recover the relative positions of camera in each of the 2 views under some conditions.

2.2.1 Epipolar constraints in 3D coordinates

In Figure 2, The epipolar line $(e'p')$ can be seen as a projection of the ray going from the object P to the image point p taken by the camera placed in O_1. Conversely, the line (ep) is the projection of the ray arriving to the second image. This can be seen equivalently as intersection the the plane generated by O_1, O_2 and p with the second image plane on the epipolar line $(e'p')$. A second constraint can be given on the epipolar line (ep).

The constraints can be written by an algebric condition. Let's suppose that $\widetilde{X}_c$ is the position of the object P in the coordinate of O_1 and $\widetilde{X}'_c$ its position in the coordinate system of O_2. Let's denote R and T the relative rotation and translation between the two cameras and $T_\times$ the cross product matrix of vector T defined by equation 8. The relation between the two system coordinates is defined by:

$$\widetilde{X}_c = R\widetilde{X}'_c + T \quad (7)$$

$$T_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \quad (8)$$

Hence,

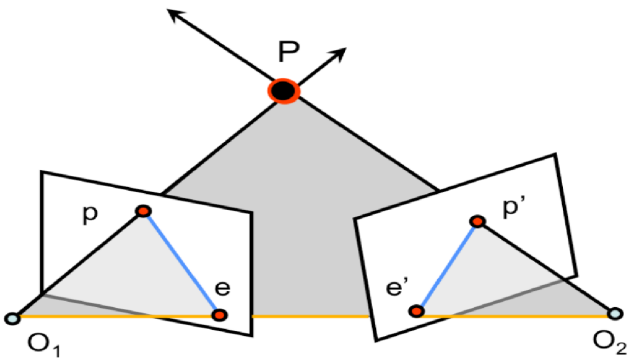

Figure 2. Epipolar geometry.

$$\widetilde{X}_c^T T_\times R\widetilde{X}'_c = \widetilde{X}_c^T T_\times(\widetilde{X}_c - T) = \widetilde{X}_c^T T_\times \widetilde{X}_c - \widetilde{X}_c^T T_\times T = 0 \quad (9)$$

The first term is null because

$$X^T T_\times X = t_y X_c Z_c - t_z X_c + t_z X_c Y_c - t_x Y_c Z_c + t_x Y_c Z_c - t_y X_c Z_c \quad (10)$$

while the second is null because it represents a cross product of T with itself.

Consequently, we can define the essential matrix $E \equiv T_\times R$ and the epipolar constraints are reduced to the algebric condition:

$$\widetilde{X}_c^T E \widetilde{X}'_c = 0 \quad (11)$$

2.2.2 Constrains in 2D coordinates system

The constraint 11 can be expressed in the image coordinate system using calibration matrices K and K'. The link between these two coordinates is shown in equation 5. Therefore, the constraint 11 can be expressed also as:

$$\begin{aligned} 0 &= \widetilde{X}_c^T E \widetilde{X}'_c \\ &= (K^{-1}\widetilde{u})^T E (K'^{-1}\widetilde{u'}) \\ &= \widetilde{u}^T (K^{-1T} E K'^{-1}) \widetilde{u'} \\ 0 &= \widetilde{u}^T F \widetilde{u'} \text{ where } F \equiv K^{-1T} E K'^{-1} \end{aligned} \quad (12)$$

F is known as the fundamental matrix. It is a 3×3 matrix that has rank 2 since thee epipolar $\widetilde{e}$ is a non null vector of its null space.

2.3. Matching points

A fundamental step of structure from motion is being able to identify 2D point correspondences in two or more views that refer to the same 3D point. To do so, we will use feature matching methods like SIFT. Sift works both as keypoint detector and as local descriptor. In SIFT, we uses Laplacian of Gaussians computed with various σ values which will act as blob detector that detects blobs in different sizes. In practice, keypoints are found by searching for local maxima across the space and the scale.

Then we can compute features on these points of interest. A 16x16 neighbourhood is taken around each keypoint. It is divided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin gradient orientation histogram is created and filled with gradient magnitude value. So a total of 128 bin values are available. It is represented as a flat vector to form keypoint descriptor. This descriptor will be the key to keypoints matching.

Since SIFT descriptors are invariant to scale and rotation, we can search for the nearest neighbor for each keypoint in the second view and obtain matching points.

3. Methodology

3.1. From 2 views to point cloud

Given two views, the coordinate system of one view is considered as the reference. Using matched points between images, we can first estimate the fundamental matrix and deduce the essential matrix. It is possible then to recover rotation and translation of a camera with respect to view fixed as reference. Finally, we can find 3D coordinate of a view in the coordinate system of the reference camera using triangulation.

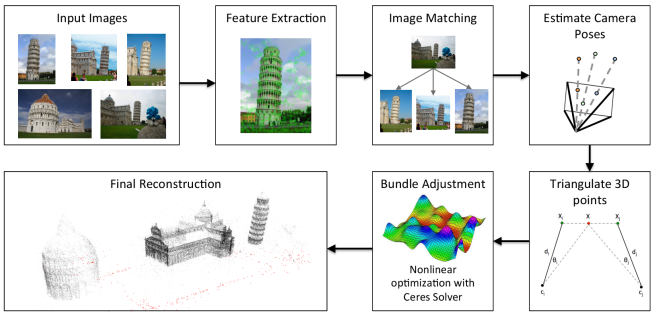

Figure 3. All steps in SFM.

3.1.1 Matching key points from 2 views

Using SIFT descriptors described in 2.3, several points in the two image coordinate systems are matched. The strategy to achieve confident matches is to do at a first place a 2 nearest neighbors match between descriptors extracted from two the two images. Secondly, we only keep matches that are not ambiguous i.e. where distance between SIFT feature from view 1 and its second nearest neighbor in view 2 is less than some ratio of the distance from its closest neighbor in view 2. We fixed the ratio at 0.75 in our experiment. An illustration of this step is given in Figure 4

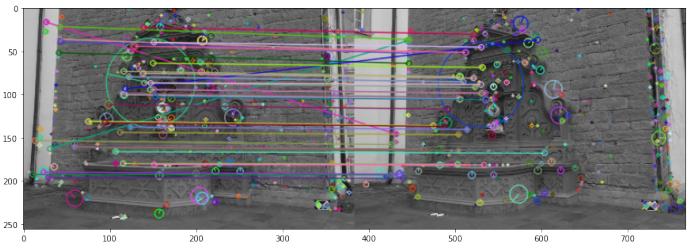

Figure 4. Example of matching key points from 2 views.

3.1.2 Estimation of Projection matrices

In section 2.2.2, relation between image coordinates and fundamental matrix is explicated. Using the n matched point between two views one can find the matrix F by solv-

ing the system:

$$Af = 0 \tag{13}$$

$$A = \begin{bmatrix} u'_1u_1 & u'_1v_1 & u_1 & v'_1u_1 & v'_1v_1 & v_1 & u_1 & v_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u'_nu_n & u'_nv_n & u_n & v'_nu_n & v'_nv_n & v_n & u_n & v_n & n \end{bmatrix} \tag{14}$$

$$f = \begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{21} & f_{22} & f_{23} & f_{31} & f_{32} & f_{33} \end{bmatrix}^T \tag{15}$$

Since the point correspondences are computed using SIFT, the data is likely to be noisy and contains several outliers in general. Thus, to remove these outliers, we use RANSAC algorithm to obtain a better estimate of the fundamental matrix. So, out of all possibilities, the F matrix with maximum number of inliers is chosen.

The next step is to find the essential matrix E using calibration matrices of the two views, since F is define by equation 12. In our experiment, the same camera is used, so we apply the same calibration matrice to recover E up to a scale:

$$E \sim K^T F K \tag{16}$$

E has the same rank as matrix F which is equal to 2. So E has 0 as singular value with a multiplicity equal to 1. Also by definition of E, we can show that $E^T E = T_\times^T T_\times$ has a non null eigenvalue with multiplicity equal to 2. Since E is defined up to a scale we can consider that the singular values of E are $(1, 1, 0)$. Hence we can deduce the following equations:

$$E = U^T \Sigma V \tag{17}$$

$$T_\times \sim U^T \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} U \tag{18}$$

$$R = V^T \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V \tag{19}$$

Scale is fixed such that $|T| = 1$ and signs are chosen such that the points lies in front of the camera. This allows to recover the projection matrices: P for the reference arbitrary chosen view and P' for the second view such that:

$$P = K[I, 0] \tag{20}$$

$$P' = K[R, T] \tag{21}$$

3.1.3 Triangulation and point cloud

To find 3D coordinate of an image point we solve the following optimisation problem for every point matched in the 2 views:

$$\mathbf{X} = \arg\min_{\mathbf{X}} ||u - \hat{u}(P, \mathbf{X})|| + ||u' - \hat{u'}(P', \mathbf{X})|| \tag{22}$$

Applied to each match, this operation results in building a point cloud of all the identified matches with coordinates in the 3D system of the camera chosen as reference. A simple strategy to solve this problem is to use cross product to get rid of the scale factor. In fact we have:

$$\hat{u} = \alpha P\mathbf{X} \qquad \text{and} \qquad \hat{u'} = \alpha' P'\mathbf{X} \tag{23}$$

where $\hat{u}$ and $\hat{u'}$ are matching points of the same 3D point $\mathbf{X}$. Since $\hat{u}$ and $P\mathbf{X}$ are collinear we have:

$$\hat{u} \times P\mathbf{X} = 0 \tag{24}$$

This can be written as:

$$x(\mathbf{p^{3\top}}\mathbf{X}) - (\mathbf{p^{1\top}}\mathbf{X}) = 0 \tag{25}$$

$$y(\mathbf{p^{3\top}}\mathbf{X}) - (\mathbf{p^{2\top}}\mathbf{X}) = 0 \tag{26}$$

$$x(\mathbf{p^{2\top}}\mathbf{X}) - y(\mathbf{p^{1\top}}\mathbf{X}) = 0 \tag{27}$$

$$\tag{28}$$

where $\hat{u} = [x, y, 1]$ and $\mathbf{p^{i\top}}$ is the i^{th} row of P. An equation of the form $AX = 0$ of the image points can be formed with: A= $\begin{bmatrix} x(\mathbf{p^{3\top}}) - (\mathbf{p^{1\top}}) \\ y(\mathbf{p^{3\top}}) - (\mathbf{p^{2\top}}) \\ x'(\mathbf{p^{3\top}}) - (\mathbf{p^{1\top}}) \\ y'(\mathbf{p^{3\top}}) - (\mathbf{p^{2\top}}) \end{bmatrix}$

This can be seen as minimizing $||AX||^2$ st. $||X|| = 1$ which can be solved by SVD decomposition of A and considering the eigenvector with the smallest eigenvalue.

3.2. Point cloud from multiple views

The methodology we described in the previous section allows us to construct a point cloud from two images sharing a number of matching keypoints. But this is not enough when we want to do Multiple-view structure from motion. When dealing with multiple images we can do reconstruction sequentially by incorporating successive views one at a time. At each step, a partial reconstruction is extended by computing the positions of all 3D points that are visible in two or more views.

3.2.1 Merging partial reconstructions

Our first method consists in merging partial reconstructions obtained from 2 views using corresponding 3D points. Typically, two-view reconstructions are obtained using adjacent image pairs; then they are merged using corresponding 3D points. The merging step will consist in adjusting point clouds knowing some matching points from the two clouds by finding the rotation matrix R and the translation vector between the two views. Let $P_{1..n}$ and $P'_{1..n}$ be a set of 3D matching points. We want to find R and t that minimize the mean square error $f(R, t) = \sum_{i=1}^{n} \|P_i - (RP_i + t)\|^2$.

Let's first consider the barycenters of these two clouds: $P_m = \sum_{i=1}^{n} P_i$, $P'_m = \sum_{i=1}^{n} P'_i$ and the centered points $Q_i = P_i - P_m$ and $Q'_i = P'_i - P'_m$.
We compute the gradient of the mean square error with respect to t:

$$\frac{\partial f}{\partial t}(R,t) = -2\sum_{i=1}^{n}(P_i - RP'_i - t) \tag{29}$$

Setting this gradient to 0 will give us:

$$t_{min} = P_m - RP'_m \tag{30}$$

At minimum, if it exists, f can be rewritten (using $RR^T = I$):

$$f(R,t_{min}) = \sum_{i=1}^{n} \|Q_i - RQ'_i\|^2 \tag{31}$$

$$= \sum_{i=1}^{n} Q_i Q_i^T + Q'_i {Q'_i}^T - 2{Q_i}^T R Q'_i \tag{32}$$

Since the two first terms of the sum don't depend on R we can simply maximize the function g defined as:

$$g(R) = \sum_{i=1}^{n} {Q_i}^T R Q'_i = \sum_{i=1}^{n} Tr(RQ'_i {Q_i}^T) = Tr(RH) \tag{33}$$

where $H = \sum_{i=1}^{n} Q'_i {Q_i}^T$. We use the Singular Value Decomposition (SVD) of H: $H = U\Sigma V^T$ and rewrite g:

$$g(R) = Tr(RH) = Tr(RU\Sigma V^T) = Tr(\underbrace{(V^T RU)}_{W}\Sigma) \tag{34}$$

Now, we use a result from Whaba theorm:
Given a positive diagonal matrix Σ, there exists a rotation matrix W that maximizes $Tr(W\Sigma)$ which is the identity. Since $W = V^T RU$ is a rotation matrix ($WW^T = I$), we have $W = I$. In conclusion, the solution to this problem is:

$$R = VU^T \qquad \text{and} \qquad t = P_m - RP'_m \tag{35}$$

To our knowledge this method is not used in classical SFM algorithm or at least it is not the most common method used but it is used in Iterative closest point (ICP) algorithm for adjusting point clouds. We implement this method and improve it by using RANSAC algorithm to eliminate outliers because the point correspondences are noisy and contain false matched points.In practice, we estimate R and t using eq 35 for many sub-samples of all matched keypoints iteratively and we select R and t yielding the smallest mean squared error.

3.2.2 Resection

An alternative method is to determine the pose of each additional view using already-reconstructed 3D points. We start with a point cloud formed using two views and we add sequentially new points from new views. Since we have a set of n 3D points in the world, their 2D projections in the image and the intrinsic camera parameter, the 6 degree of freedom camera pose can be estimated using linear least squares. This fundamental problem, in general is known as Perspective-n-Point (PnP). A commonly used solution to the problem exists for n = 3 called P3P, and many solutions are available for the general case of n 3. We use the efficient PnP (EPnP) method which was developed by Lepetit, et al. In the implementation, we use the opencv RANSAC version of this algorithm.

4. Experiments

In this section, we will present some results of reconstruction we obtained using our implementation of SFM. As discussed in the previous section, we proposed two different methods for dealing with multiple views: the resection method and the merge method. We implemented both of them in python. We will also compare our implementation to existing solutions like VisualSFM.

VisualSFM is a GUI application for 3D reconstruction using structure from motion (SFM). It is able to run very fast by exploiting multicore parallelism in feature detection, feature matching, and bundle adjustment. It includes many other features like 3D dense reconstruction.

We will use the Fountain-P11 dataset which contains 11 different view of a fountain (Figure 5) and for which we know the intrinsic parameters of the camera.

Figure 6 shows results obtained with the merge method. We colored the point cloud using values from pixels in 2D and exported the result in .ply file and used CloudCompare software to visualize results. We also show how the point cloud is sequentially augmented by adding one view at a time.

Figure 7 shows reconstruction results using the second method. Results using the resection method were slightly better with less noise and sharper edges. It is still a simple reconstruction method that can be improved using for example bundle adjustment which comes usually as a final step in SFM. At this stage, we refine the poses and 3D points together, initialized by previous reconstruction by minimizing a reporjection error.

To see what more advanced SFM implementations could achieve, we use the VisualSFM application on the Fountain-P11 dataset. Figure 8 shows the 3D reconstruction obtained with VisualSFM using SFM and bundle adjustment. Unfortunately, we cannot export the point cloud to visualize it in CloudCompare, but from the 2D image of the obtained

Figure 5. Images from Fountain-P11 dataset

point cloud, we can see that we have obtained quite good results even though our method is simple (no bundle adjustment step). VisualSFM point cloud have less points than our point cloud even though it extracted more keypoints using SIFT. We think that they do more filtering and they retain only good 3D points later (for eg. points appearing in more than k views). This can explain why we have a denser but more noisy pointy cloud. We obtained 103k point using the merging method and 78k using the resection method.

We have also tried the dense 3D reconstruction algorithm (Clustering Views for Multi-view Stereo (CMVS)) included in VisualSFM which delivered outstanding results (see Figure 9).

5. Conclusion

In this project, we had the chance to review many concepts seen in the course related to stereo vision and classical computer vision algorithms for keypoint detection and image descriptors. We went through all the theoretical aspects of structure-from-motion and implemented it in python using many built in functions in open-cv. However, this method is far behind more advanced methods like CMVS for dense reconstruction which uses the output of a structure-from-motion and builds on top of it or deep learning based methods.

References

[1] Cs231a: Computer vision, from 3d reconstruction to recognition. `http://web.stanford.edu/class/cs231a/course_notes.html`.

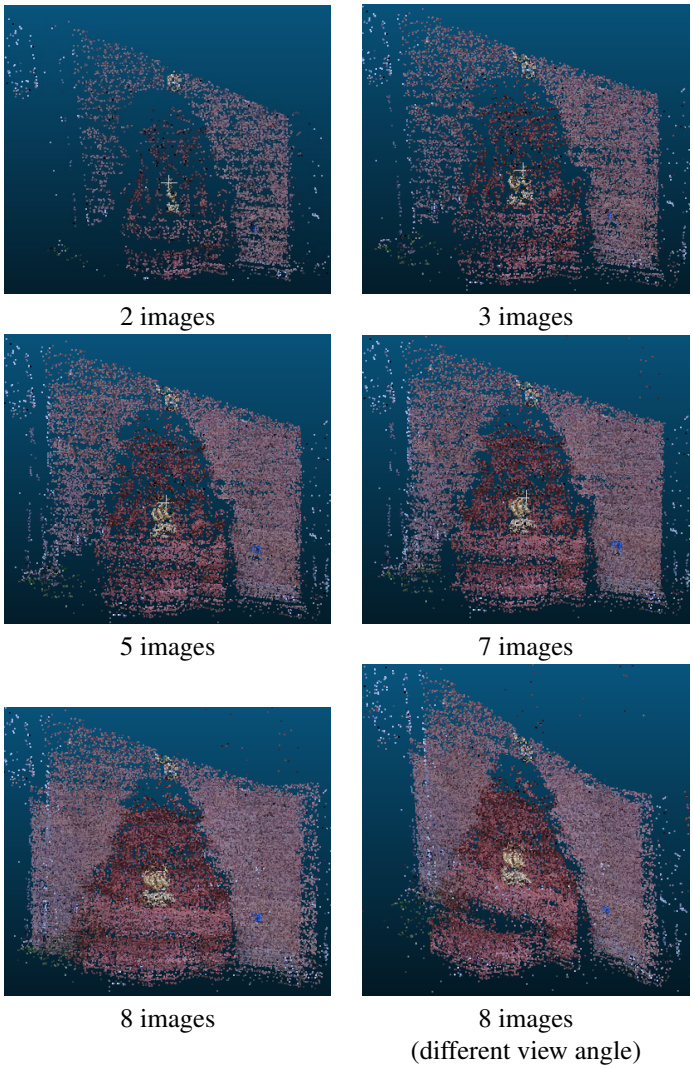

Figure 6. 3D reconstruction by merging partial reconstructions

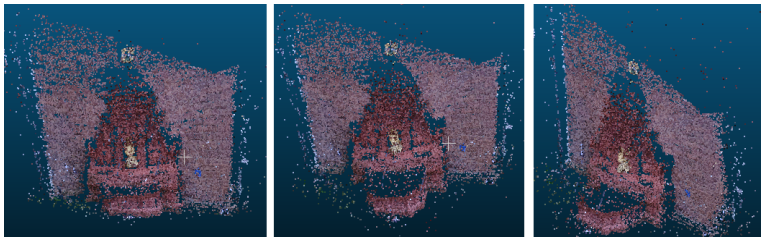

Figure 7. 3D reconstruction by Resection.

[2] M. Arie-Nachimson, S. Z. Kovalsky, I. Kemelmacher-Shlizerman, A. Singer, and R. Basri. Global motion estimation from point matches. In *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*, pages 81–88, 2012.

[3] Z. Cui, N. Jiang, and P. Tan. Linear global translation estimation from feature tracks. *CoRR*, abs/1503.01832, 2015.

[4] Z. Cui and P. Tan. Global structure-from-motion by similarity averaging. In *2015 IEEE International Conference on*

Figure 8. 3D reconstruction using VisualSFM (SFM+Bundle adjustment).

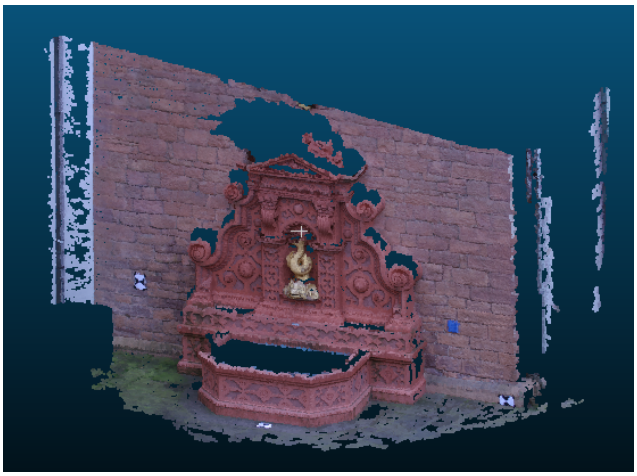

Figure 9. 3D dense reconstruction using VisualSFM.

Computer Vision (ICCV), pages 864–872, 2015.

[5] O. Enqvist, F. Kahl, and C. Olsson. Non-sequential structure from motion. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 264–271, 2011.

[6] V. Govindu. Combining two-view constraints for motion estimation. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 2, pages II–II, 2001.

[7] V. Govindu. Lie-algebraic averaging for globally consistent motion estimation. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I, 2004.

[8] https://phortail.org/club-informatique/definition-informatique 136.html. Date de dernière consultation : 26/08/2019.

[9] N. Jiang, Z. Cui, and P. Tan. A global linear method for camera pose registration. In *2013 IEEE International Conference on Computer Vision*, pages 481–488, 2013.

[10] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[11] P. Moulon, P. Monasse, and R. Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *2013 IEEE International Conference on Computer Vision*, pages 3248–3255, 2013.

[12] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, jul 2006.