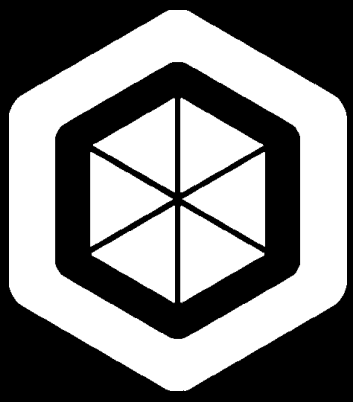

LC-MC0415

LABORATORY OF CERTIFICATION
CERTIFICATION OF MODULES

July 15, 2015

N. Hamel
Deputy Director, FOXEL Labs Research Center
Laboratory of Certification

FOXEL Labs, Laboratory of Certification

Contents

1 Introduction

In the context of big data, it is necessary to define models for processing time, memory consumption and storage space. FOXEL being a significant generator of numerical data through its photogrammetry solutions, mathematical models are defined here to describe those quantities according to FOXEL software and external solutions.

Models are defined for each steps of the data processing and an overall model is then deduced. The different steps are briefly described in terms of parallel processing in order to justify the considered mathematical models that are kept as simple as possible.

Since this certification procedure occurs early in the modules establishment procedure, processing modules appear more as processing steps rather than well identified and standardized processing modules.

2 Models definition

Simple mathematical models are defined here to describe the processing time, the memory consumption and the storage space of each processing steps. The chosen models depend on the way the different steps are parallelized.

In order to keep very simple models, only two parameters are considered as models variables. The first variable is the considered number of camera captures. Considering Eyesis4π camera device, a capture is defined as 26 images that induce a single panorama.

The second variable is the number of threads that are available for data processing. Moreover, an assumption is made about the available threads : one thread is assumed to have its very own processor core to perform its task. It follows that threads are not identify with processors as they are defined by the hyper-threading technology.

2.1 Digitization data

Formally, digitization data is not related to a processing step and so is not related to processing time or memory consumption models. Nevertheless, a storage model is here defined.

To describe the digitization storage model, a simple linear model is defined as follows :

$$s_d(n) = 2\gamma_d n \tag{1}$$

where n gives the number of camera captures, γ_d being the model parameter and gives the storage cost of a single camera capture. Data coming from the digitization are concatenated in single files that are then slitted to obtain the image files, explaining the factor 2. Keeping both sets of files is a question of redundancy.

2.2 Equirectangular projections

From the digitization files, after splitting, equirectangular projections of each camera sensor are computed [1] and constitute the first step of the data processing. During this step, optical distortions and chromatic aberrations are removed. The processing of one camera capture is totally independent of any other capture processing. It follows that

captures processing can be fully parallelized.

Considering the processing time model and taking into account the parallelization possibility, the following model is defined :

$$t_e(n,p) = n\left(\frac{\alpha_{e,1}}{p} + \alpha_{e,2}\right) \tag{2}$$

where n gives the number of considered captures when p gives the number of available threads. A linear component in n is added due to considerations on data traffic.

The memory consumption model is a linear model that only depends on the number of considered threads for captures processing. It is defined as :

$$m_e(p) = \beta_e p \tag{3}$$

where p is the number of threads and β_e is the model parameter.

This processing induce equirectangular projection image files for each sensor of each capture. The storage model for this step then follows the same form as the previously defined storage model :

$$s_e(n) = \gamma_e n \tag{4}$$

where n is the number of processed captures.

2.3 Rectilinear projections

The rectilinear projections computation [2] consists in obtaining the sensors images without optical distortions and chromatic aberrations on the basis of the equirectangular projections and through a gnomonic projection. As the previous step, this process can be fully parallelized.

It follows that the processing time model is given by a similar relation as previously :

$$t_r(n,p) = \alpha_r \frac{n}{p} \tag{5}$$

where n is the captures number and p the threads count.

The storage model is then as defined for the computation of the equirectangular projections, that is a simple linear model :

$$s_r(n) = \gamma_r n \tag{6}$$

where n is the number of processed captures.

2.4 Panoramas assembly

The panoramas assembly [3] takes the equirectangular projections of each sensors and stitch them together to obtain the entire panorama equirectangular projection for each camera capture. Again, according to captures, this process is totally parallelized.

It follows that the same processing time model is defined for this processing step as the previous one :

$$t_a(n,p) = \alpha_a \frac{n}{p} \tag{7}$$

with n and p the captures and the threads count. The memory consumption model is then given by the following relation :

$$m_a(p) = \beta_{a,1} p + \beta_{a,0} \tag{8}$$

considering a linear model according to thread count p. The storage model is also identical to the previous processing steps :

$$s_a(n) = \gamma_a n \tag{9}$$

with n the number of considered captures. For a single capture, stereo pairs not being considered in panoramas stitching, 24 images are considered as input when only one image is given as output, but the model remains linear according to n.

2.5 Poses estimation

The pose estimation [4] is responsible of estimating the position and orientation of each sensor of the camera device. It is decomposed it three major sub-steps that are images matching, sensor poses estimation and the refinement of the poses estimation. In this case, the parallelization of the each sub-steps has it own constraint and have to be treated separately.

Starting with the image matching, features are detected in each image before starting the image comparison for matching. It means that already two components appears in the processing time model, one in $O(n)$ and one in $O(n^2)$. This part can be fully parallelized.

The estimation of the sensors poses can be described by in a $O(n^2)$ model and is partially parallelized. Finally, the refinement of the estimated poses can be described by a $O(n\log(n))$ model and is also partially parallelized.

In the overall, the processing time model for this steps can be written as follows :

$$t_s(n,p) = \frac{n\left(\alpha_{s,3}n^2 + \alpha_{s,2}n + \alpha_{s,1} + \alpha_{s,b}\log(n)\right)}{p} \tag{10}$$

Models are not define for poses estimation about memory consumption and storage. Considering memory consumption, the software is designed in such a way that it takes part of the available memory on the considered computer. It follows that memory consumption for poses estimation do not depends on captures count or threads number.

Considering storage model, it appears that results storage load is mainly due to the *keypoint* and match files, the output point cloud being negligible in terms of size according to the previous steps, and their size is highly depending on the scene captured by the camera. At this point, it is too complicated to define a mathematical model for storage of this processing step.

2.6 Densification

The densification [5] is the final step for digitized environment reconstruction. It considers the estimated sensor poses and their rectilinear projection images to build a point cloud expected to be as dense as possible.

As for poses estimation, two main sub-steps are performed during densification that are feature detection and images matching. It follows that a processing time model can be defined as follows :

$$t_f(n,p) = \frac{\alpha_{f,2}n^2 + \alpha_{f,1}n}{p} \tag{11}$$

where n is the number of captures and p the threads count.

As for pose estimation, neither memory consumption model and storage model are defined for densification. The assumptions made for poses estimation remaining valid for the densification part.

Because this software is not maintained by FOXEL, the defined model is less reliable than the previous defined models.

2.7 Data models : Summary

For the overall processing, from digitization to metric reconstruction environment, the processing time model is given by :

$$\begin{aligned} t(n,p) &= t_e(n,p) + t_r(n,p) + t_p(n,p) + t_s(n,p) + t_f(n,p) \\ &= \frac{n}{p}\left[\alpha_{e,1} + p\alpha_{e,2} + \alpha_r + \alpha_a + \alpha_{s,1} + \alpha_{f,1} + n\left(\alpha_{s,2} + \alpha_{f,2} + n\alpha_{s,3}\right) + \log(n)\alpha_{s,b}\right] \end{aligned} \tag{12}$$

the memory consumption model is given by :

$$\begin{aligned} m(n,p) &= m_e(p) + m_r(p) + m_a(p) \\ &= p\left(\beta_e + \beta_{a,1}\right) + \beta_{a,0} \end{aligned} \tag{13}$$

and finally, the storage cost model is summarized by :

$$\begin{aligned} s(n) &= s_d(n) + s_e(n) + s_r(n) + s_a(n) \\ &= n\left(2\gamma_d + \gamma_e + \gamma_r + \gamma_a\right) \end{aligned} \tag{14}$$

It follows that the data processing can be describe through a 17 parameter simple model.

3 Models parameters estimation

In order to compute the 17 parameters of the models, a entire server is immobilized for specific measures. The server is composed of four physical Intel® Xeon® CPU E5-4657L v2 processor with 24 threads for 12 cores at 2.40 GHz and offers 256 Gb of memory. According to the assumption made on threads, the maximum number of available threads is then 48.

3.1 Digitization data

According to the defined model (1), the digitization data storage model requires the computation of only one parameter. It is deduced through a simple linear regression of a collection of measures. In this case, the parameter is :

$$\gamma_d = 1.06227199 \times 10^{-2} \tag{15}$$

The figure 1 gives an illustration of the storage model of digitization data according to camera captures count.

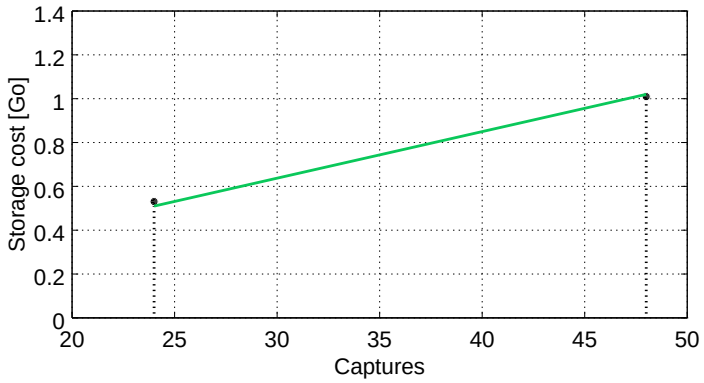

Figure 1: Digitization data storage model

3.2 Equirectangular projections

In order to determine the four parameters of the models for the computation of the equirectangular projections, a set of 24 and 48 camera captures is considered and pro-

cessed with 1, 6, 12, 24 and 48 threads.

According to those measures, the processing time parameter is computed through the linear system resolution. The memory consumption parameter is computed through linear regression on measures as the storage model parameter.

The following parameters are obtained :

$$\alpha_{e,1} = 2187.365358,\ \alpha_{e,2} = 266.344152,\ \beta_e = 3.904443,\ \gamma_e = 0.674895 \tag{16}$$

The figure 2 gives an overview of the data processing time model according to number of camera captures and the threads count.

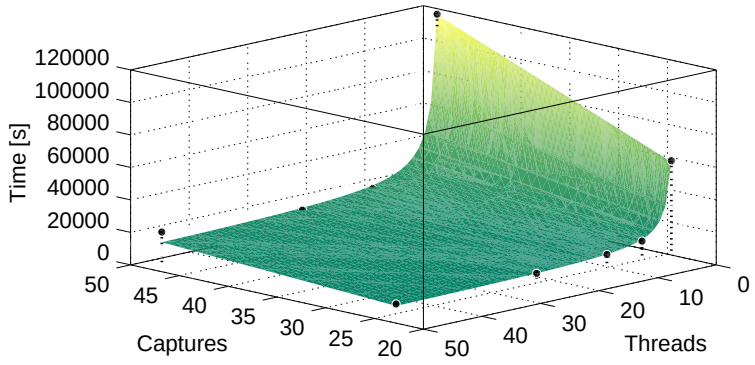

Figure 2: Equirectangular projections time model

The figure 3 gives an illustration of the memory usage model when the figure 4 gives the storage model representation according to camera captures count.

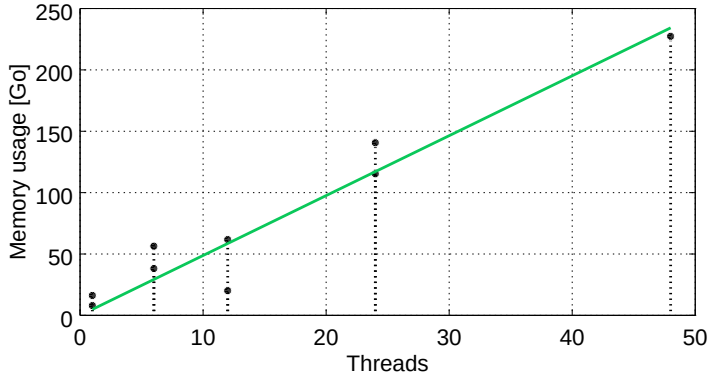

Figure 3: Equirectangular projections memory model

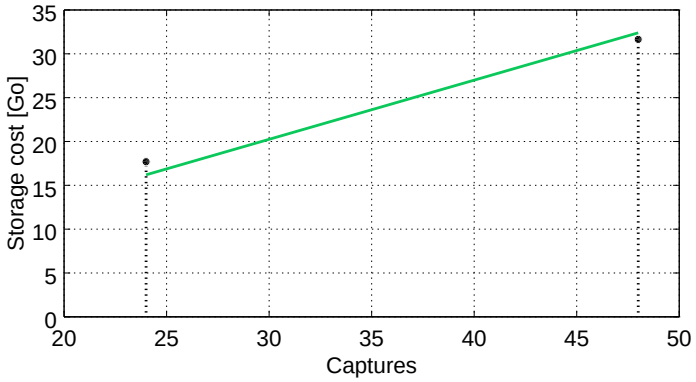

Figure 4: Equirectangular projections storage model

3.3 Rectilinear projections

The computation of the rectilinear projections models parameters is done through linear regression on the performed measures. Those measures are performed using the same captures and threads count as for equirectangular projections step. The determined models parameters are summarized by :

$$\alpha_r = 71.733030\,s,\ \beta_r = 0.367728,\ \gamma_r = 0.396186 \tag{17}$$

The figure 5 gives a representation of the processing time according to threads and camera captures when the figure 6 gives it for storage model.

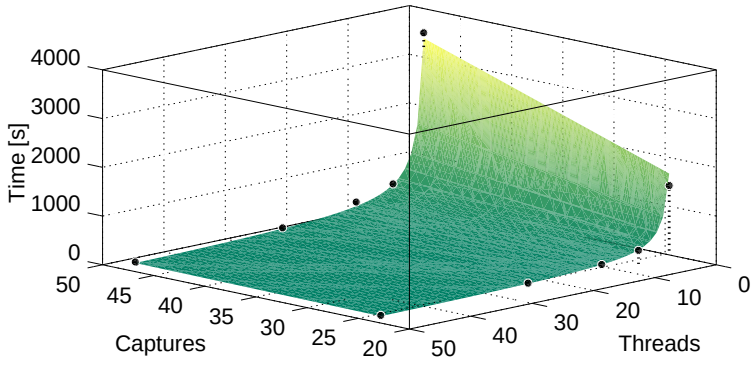

Figure 5: Rectilinear projections time model

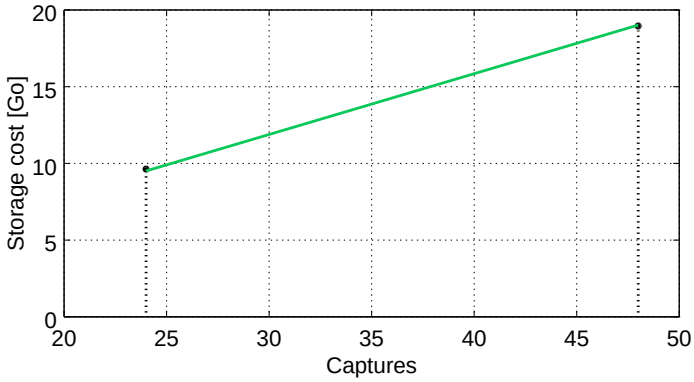

Figure 6: Rectilinear projections storage model

3.4 Panoramas assembly

The computation of the panoramas assembly models parameters is identical to the previous processing steps considering the same amount of captures and threads for measures. The models parameters are given by :

$$\alpha_a = 188.119286,\ \beta_{a,1} = 0.336653,\ \beta_{a,0} = 12.465222,\ \gamma_a = 0.272727 \qquad (18)$$

The figure 7 gives the extrapolated model for panorama assembly processing time according to camera captures and threads.

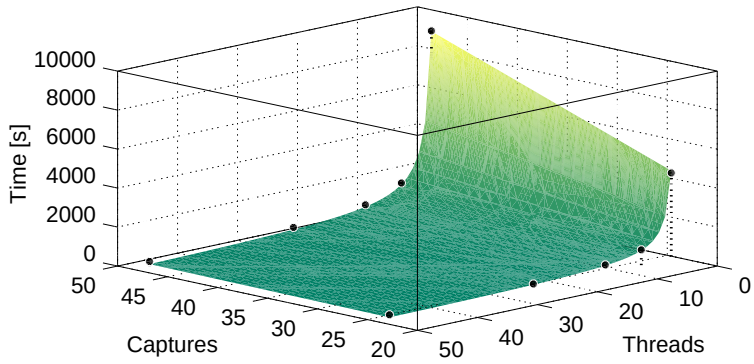

Figure 7: Panoramas assembly time model

The figure 8 gives an overview of the memory consumption model according to threads count.

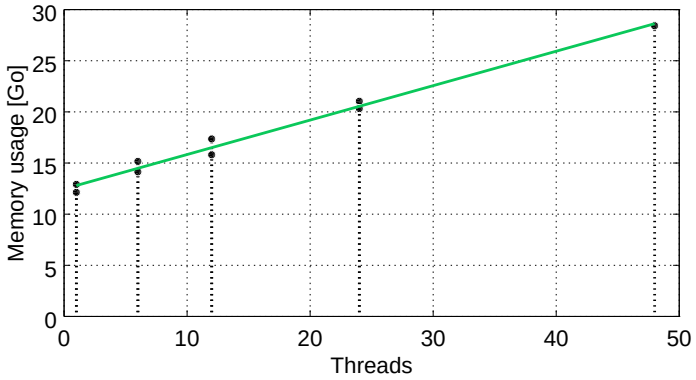

Figure 8: Panoramas assembly memory model

The figure 9 gives the estimated storage model according to camera captures count.

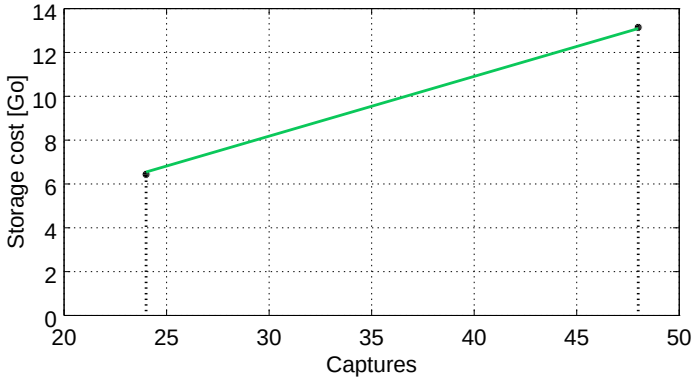

Figure 9: Panoramas assembly storage model

3.5 Poses estimation

The model for the poses estimation involve 4 parameters that are computed using measures on data sets of 12, 24 and 48 camera captures and for each cases considering 6, 24 and 48 threads. The parameters of the processing time model are computed through a linear system resolution.

The computed parameters are summarized by :

$$\alpha_{s,3} = 2.841721,\ \alpha_{s,2} = 169.227659,\ \alpha_{s,1} = 10.574262,\ \alpha_{s,b} = 31.163540 \qquad (19)$$

The figure 10 gives an representation of the processing time according camera captures and threads count.

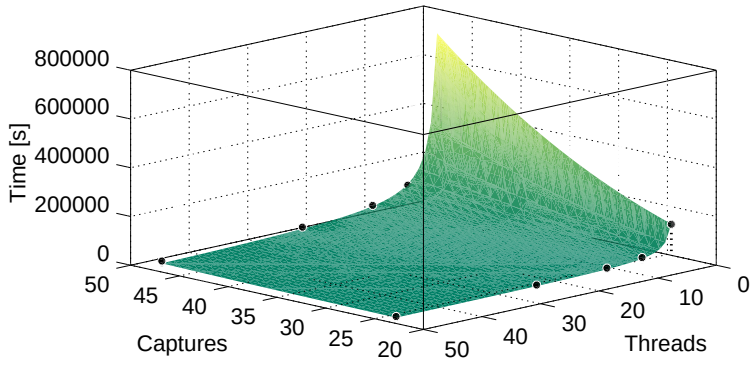

Figure 10: Poses estimation time model

3.6 Densification

The determination of the model parameters for the densification processing step is identical as the parameters computation of poses estimation steps using the same amount of captures and threads. The computed parameters are :

$$\alpha_{f,2} = -6.808665,\ \alpha_{f,1} = 493.008166 \tag{20}$$

The figure 11 gives an overview of the processing time of the densification step according to camera captures and threads.

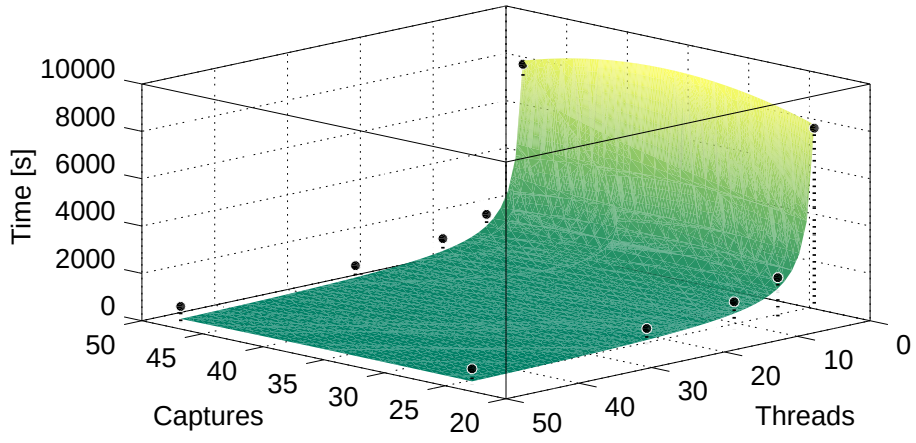

Figure 11: Densification time model

As for poses estimation, the results and so the processing is highly depending on the captured scene. In our case, it seems that for both 24 and 48 captures, the same amount of them have been kept by the poses estimation process, the other having been removed by the algorithm. It explains why the measure have this profile.

As a consequence, the model parameter for processing time should be considered with precaution according to captures count. The profile according to thread count seems clearly better.

4 Conclusion

The defined data models are the first precise description of FOXEL data processing time, memory consumption and storage cost. Due to the complexity of the overall models, it follows that models accuracy is not fully validated and models precision will increase with time.

Taking into account that performing the measures able to determine accurate models parameters induce high computation load, one has to keep in mind that the parameters given through this certification have to be considered as a first estimation.

Moreover, the equirectangular projections and the densification steps are performed through software not maintained by FOXEL. It follows that the models, especially for processing time, is determined through assumptions on their algorithmic nomenclature.

Nevertheless, this certification is able to fulfil the required computation of processing time, memory consumption and storage necessities according to the FOXEL photogrammetric solutions.

Going toward industrial photogrammetric solutions, the different certified steps are called to become standardized module of processing. It means that the required measures to determine and certify those mathematical models will be natively handled by the module itself, bringing this certification procedure toward automation.

Reference

[1] *imagej-elphel*; software available at
https://github.com/Elphel/imagej-elphel/tree/master

[2] *gnoproj 1.1*; software available at
https://github.com/FoxelSA/gnoproj/releases/tag/v1.1

[3] *enblend/enfuse*; software available at
http://enblend.sourceforge.net/

[4] *openMVG 0.8.1*; software available at
https://github.com/openMVG/openMVG/releases/tag/v0.8.1

[5] *PMVS-2*; software available at
http://www.di.ens.fr/pmvs/

Acknowledgement

K. Velickovic
Developer, FOXEL

Acknowledgement to reviewers

P. Moulon
Director, Laboratory of Photogrammetry, FOXEL Labs

S. Flotron
Research Fellow, Laboratory of Photogrammetry, FOXEL Labs