Efficient Camera Smoothing in Sequential Structure-from-Motion using Approximate Cross-Validation

Michela Farenzena, Adrien Bartoli, and Youcef Mezouar

LASMEA, UMR6602 CNRS
Université Blaise Pascal, Clermont, France
Michela.Farenzena@univ-bpclermont.fr

Abstract. In the sequential approach to three-dimensional reconstruction, adding prior knowledge about camera pose improves reconstruction accuracy. We add a smoothing penalty on the camera trajectory. The smoothing parameter, usually fixed by trial and error, is automatically estimated using Cross-Validation. This technique is extremely expensive in its basic form. We derive Gauss-Newton Cross-Validation, which closely approximates Cross-Validation, while being much cheaper to compute. The method is substantiated by experimental results on synthetic and real data. They show that it improves accuracy and stability in the reconstruction process, preventing several failure cases.

1 Introduction

The sequential approach to Structure-from-Motion (SfM) [1–4] entails starting from a seed reconstruction, then adding one new view at a time, updating the structure accordingly. The strategy that is usually adopted to robustly calculate a new camera pose is to use the already estimated three-dimensional (3D) points to solve a resection problem [1, 5, 6] within RANSAC [7]. In our experience however, this does not guarantee a good initialisation for bundle adjustment and does not prevent the reconstruction process from failing. Resection indeed uses only local information; it is prone to drifting and local instabilities.

It is commonly admitted that using prior knowledge improves the quality of an estimate. In video sequences, it is reasonable to add a continuity or smoothing prior on the camera trajectory, encouraging each camera to lie close to the previous ones. We minimize a compound cost function, which sums the reprojection error and a smoothing penalty. The trade-off between these is regulated by the *smoothing parameter*.

The smoothing parameter is commonly tuned by trial and error, and is kept constant in the whole sequence. We show that accuracy can be enhanced by choosing this parameter automatically, customising the smoothness for each pose. The idea is to estimate the *most predictive camera pose*, in the sense that it can 'explain' the whole image as well as possible, given a restricted set of data points. This is a typical machine learning problem. Cross-Validation (CV) techniques can be used. The dataset is split in a training and a test set. The smoothing parameter for which the trained model minimizes the test error is selected.

The main drawback of CV is the computational cost. The greedy formula to compute the leave-one-out CV score (CV_{loo}) for one certain value of λ requires to solve n nonlinear least squares problems, with n the number of points in the dataset. For the case of regular linear least squares there is a simple non-iterative formula that gives the CV_{loo} score without having to solve as many problems as there are measurements [8]. We derive a similar non-iterative formula for the nonlinear least squares resection problem. We define the Gauss-Newton CV_{loo} (GNCV_{loo}) score and show that it closely approximates the true CV_{loo}. The computation of GNCV_{loo} requires to solve only *one* nonlinear least squares problem. This makes the estimation of the smoothing parameter a much cheaper problem. Thus, our method could be embedded in other SfM pipelines, such as [4, 9, 10], working in real time.

The approach is validated by experimental results on synthetic and real data. They show that it increases accuracy and stability in the reconstruction process, preventing several failure cases.

2 Background on Sequential Structure-from-Motion

A common scheme for sequential SfM, in both the calibrated and uncalibrated camera cases, has three main steps. Our contribution concerns the last step.

Extraction of keyframes and keypoint matching. The first step consists in relating the images. This means extracting keypoints and matching them between the images. In video sequences the baseline (the camera displacement between two views) is however often too limited. This makes the computation of matching tensors (such as the fundamental matrix) ill conditioned. A solution is to select a subset of images (keyframes). Many ways to choose these keyframes have been proposed [2, 3, 11, 12]; they balance the baseline and the number of matched keypoints. Once keyframes are selected, matching tensors are estimated. The initial set of corresponding points is typically contaminated with outliers. Traditional least squares approaches thus fail and a robust method, such as RANSAC, must be used.

Initialisation of structure and motion. The first few views and the matched keypoints are used to retrieve a seed 3D structure of the scene and the motion of the camera. Usually two or three views are used [2, 3, 13].

Sequential processing. Keyframes are sequentially added, calculating the pose of each new camera using the previously estimated 3D points. This is a nonlinear least squares problem, called *resection*. Robust estimation is usually necessary in order to cope with outliers. Subsequently, the 3D structure is updated by triangulating the 3D points conveyed by the new view. Both structure and motion are finally adjusted using bundle adjustment, with the aim of finding the parameters for the cameras and the 3D points which minimize the mean squared distances between the observed image points and the reprojected image points.

3 Resection with Automatic Camera Smoothing

Each time a new camera is added, bundle adjustment is performed in order to refine both structure and motion. This has been proved to be the essential step to achieve a good accuracy and to prevent failures [14]. The initial estimate must however be sufficiently close to the optimal solution.

Figure 1 shows an example of reconstruction failure, from a real sequence taken by a handheld camera. At the 47th keyframe the computation stops because there are not enough points to estimate the new camera pose, meaning that all points seen in the last views have been rejected as outliers. At the moment of failure, the camera is rotating. Even if the rotation is not around the camera's optical centre, this is a delicate situation, where the field of view varies rapidly and reconstruction accuracy is crucial. It is evident from Figure 1 that the pose for the last 5 keyframes are wrongly estimated, and that bundle adjustment can not fix the problem.

3.1 A Compound Cost Function

In order to refine the initial estimate a common strategy is to minimize a data term representing geometric error, i.e. the reprojection error. The problem is formalised as a least squares minimization of the mean of squared residuals (MSR):

$$\mathcal{E}_d^2(\mathsf{P}) = \frac{1}{n}\sum_{i=1}^{n} \parallel \Psi(\mathsf{P}, \mathbf{Q}_i) - \mathbf{q}_i \parallel_2^2 \ , \tag{1}$$

where P is the projection matrix and $\mathbf{Q}_i$ is the 3D position of the image point $\mathbf{q}_i$. The function $\Psi(\mathsf{P}, \mathbf{Q}_i)$ is the reprojection of $\mathbf{Q}_i$ through P, in Cartesian coordinates. The optimal solution is usually obtained by approximating $\Psi(\mathsf{P} + \Delta, \mathbf{Q}_i) \approx \Psi(\mathsf{P}, \mathbf{Q}_i) + \mathsf{J}(\mathsf{P}, \mathbf{Q}_i)\mathtt{vect}(\Delta)$, where $\mathsf{J}(\mathsf{P}, \mathbf{Q}_i)$ is the Jacobian matrix of Ψ wrt P evaluated at $(\mathsf{P}, \mathbf{Q}_i)$, and `vect` is the operator that rearranges a matrix into a vector. The normal linear least squares equations are then solved in an iterative Gauss-Newton manner.

Since the keyframes come from a video, it is reasonable to add a smoothing penalty on the camera trajectory, saying that the position of one keyframe should not differ too much from that of the previous one. If properly weighted, this penalty increases the stability in the camera trajectory estimation.

The problem is formalised as the minimization of a compound cost function, which sums the reprojection error $\mathcal{E}_d$ and a smoothing term $\mathcal{E}_s$:

$$\mathcal{E}^2(\mathsf{P}, \lambda) = (1-\lambda)^2\mathcal{E}_d^2(\mathsf{P}) + \lambda^2\mathcal{E}_s^2(\mathsf{P}) \ , \tag{2}$$

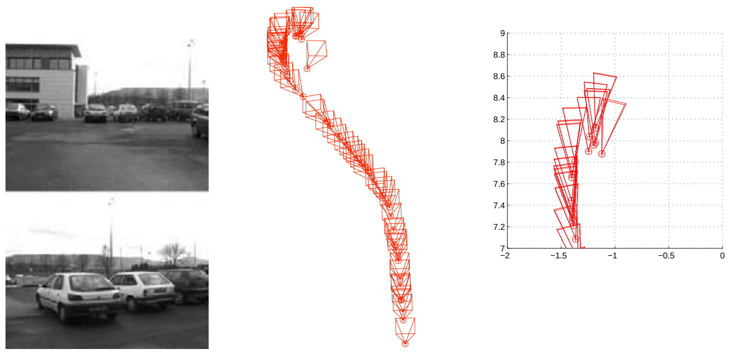
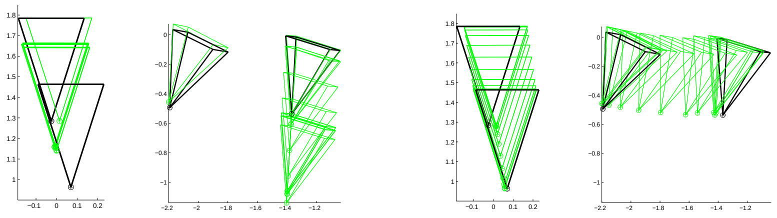


Fig. 1. Reconstruction failure in case of non smoothing of camera pose. On the left, the 1st and 47th keyframe; in the centre, a 3D view of the recovered cameras; on the right, the top view of the last keyframes. Visual inspection shows that the last five cameras are misestimated.



(a) (b) (c) (d)

Fig. 2. Examples of resection using the norm of the difference between camera matrices (a,b) or Equation (3) (c,d) as smoothing penalties. In black thick line is the ground truth for the previous and current pose; in thin green line are the poses obtained after resection, with λ varying from 0 to 1.

where λ is the *smoothing parameter*.

As a smoothness measure we use the mean of squared residuals between reprojected points in the current and previous keyframes. That is, if the two cameras are close to each other, then the points they reproject should be close as well:

$$\mathcal{E}_s^2(\mathsf{P}) = \frac{1}{n} \sum_{i=1}^{n} \| \; \Psi(\mathsf{P}, \mathbf{Q}_i) - \Psi(\mathsf{P}_p, \mathbf{Q}_i) \; \|_2^2 \;, \tag{3}$$

with P_p the projection matrix of the previous keyframe. This is actually a continuity measure, as it can be interpreted as a finite difference approximation to the first derivatives of the predicted tracks. The same scheme holds if higher order derivatives are used.

The smoothing penalty usually utilised in the literature in similar contexts is often the norm of the difference between camera matrices, but we found that the resulting cost function does not lead to the expected solution, as shown in Figure 2.

Equation (2) depends on a smoothing parameter that must be estimated. In the next section we present an automatic data-driven method solving this problem.

3.2 Smoothing Parameter Estimation by Cross-Validation

We propose to automatically determine both the camera pose *and* the smoothing parameter. The idea is to find the most predictive camera pose, in the sense that it can 'explain' the whole image, given a restricted set of 3D positions matching 2D points. This concept derives from the machine learning paradigm of supervised learning from examples.

The approach we follow is to split the data points we have in a training and a test set, and select the smoothing parameter for which the trained model minimizes the test error. A well-known method, widely applied in machine learning [16], is CV (Cross-Validation), firstly introduced in [17]. Considering that the number of samples is small, this technique recycles the test set, averaging the test error over several different partitions of the whole data set. There are different kinds of CV techniques; we chose the CV$_{loo}$ (leave-one-out CV).

The CV$_{loo}$ score is defined as a function of the parameter λ:

$$\mathcal{E}_g^2(\lambda) = \frac{1}{n} \sum_{j=1}^{n} \parallel \Psi(\hat{\mathsf{P}}_{(j)}(\lambda), \mathbf{Q}_j) - \mathbf{q}_j \parallel_2^2, \tag{4}$$

where $\hat{\mathsf{P}}_{(j)}(\lambda)$ is the camera pose estimated with all but the j-th 3D–2D point correspondence. The most predictive camera pose $\hat{\mathsf{P}}$ is obtained by solving the following nested optimization problem:

$$\hat{\mathsf{P}} = \arg\min_{\mathsf{P}} \;\; \mathcal{E}^2(\mathsf{P}, \arg\min_{\lambda} \;\; \mathcal{E}_g^2(\lambda)). \tag{5}$$

This means that the optimal camera pose is obtained by minimizing $\mathcal{E}(\mathsf{P}, \hat{\lambda})$, where $\hat{\lambda}$ is the optimal vale for λ, i.e. the one that gives the lowest CV$_{loo}$ score. $\hat{\lambda}$ is usually calculated by sampling λ over the range $[0, 1]$.

Computing the CV$_{loo}$ score using (4) is computationally expensive: it requires to solve n nonlinear least squares problems. Solving (5) is thus extremely costly.

In the next section we propose a non-iterative approximation to the CV$_{loo}$ score. Non-iterative means that it does not require to solve n nonlinear least squares problems as a trivial greedy application of Equation (4) entails.

The derivation proceeds in two steps. First we approximate the greedy formula (4) through the Gauss-Newton approximation, then we provide a non-iterative formula that exactly solves such linear least squares problems.

3.3 GNCV$_{loo}$: Gauss-Newton Leave-One-Out Cross-Validation

We rewrite Equation (2) in matrix form:

$$\mathcal{E}^2(\mathsf{P},\lambda) = (1-\lambda)^2 \frac{1}{n} \parallel \mathsf{B}(\mathsf{P}) - \mathbf{b} \parallel_2^2 + \lambda^2 \frac{1}{n} \parallel \mathsf{B}(\mathsf{P}) - \mathsf{B}(\mathsf{P}_p) \parallel_2^2 , \tag{6}$$

with $\mathsf{B}(\mathsf{P})^\mathsf{T} = [\Psi(\mathsf{P},\mathbf{Q}_1)\ \Psi(\mathsf{P},\mathbf{Q}_2)\ \dots\ \Psi(\mathsf{P},\mathbf{Q}_n)]$ and $\mathbf{b}^\mathsf{T} = [\mathbf{q}_1 \dots \mathbf{q}_n]$.
Given a certain λ, let $\hat{\mathsf{P}}_\lambda$ be the global solution to Equation (6) obtained by Gauss-Newton (GN). Let $\mathsf{C} = \mathsf{J}(\hat{\mathsf{P}}_\lambda)$ be the Jacobian matrix evaluated at $\hat{\mathsf{P}}_\lambda$. It is given by the GN algorithm. The GN approximation to $\mathcal{E}$ is:

$$\mathcal{E}^2(\hat{\mathsf{P}}_\lambda + \Delta, \lambda) \approx (1-\lambda)^2 \frac{1}{n} \parallel \mathsf{B}(\hat{\mathsf{P}}_\lambda) + \mathsf{C}\boldsymbol{\delta} - \mathbf{b} \parallel_2^2 + \lambda^2 \frac{1}{n} \parallel \mathsf{B}(\hat{\mathsf{P}}_\lambda) + \mathsf{C}\boldsymbol{\delta} - \mathsf{B}(\mathsf{P}_\mathsf{p}) \parallel_2^2 , \tag{7}$$

with $\boldsymbol{\delta} = \mathtt{vect}(\Delta)$.
Substituting $\mathbf{s} = \mathbf{b} - \mathsf{B}(\hat{\mathsf{P}}_\lambda)$ and $\mathbf{t} = \mathsf{B}(\mathsf{P}_\mathsf{p}) - \mathsf{B}(\hat{\mathsf{P}}_\lambda)$, we get:

$$\mathcal{E}^2(\hat{\mathsf{P}}_\lambda + \Delta, \lambda) \approx \tilde{\mathcal{E}}^2(\boldsymbol{\delta}, \lambda) = (1-\lambda)^2 \frac{1}{n} \parallel \mathsf{C}\boldsymbol{\delta} - \mathbf{s} \parallel_2^2 + \lambda^2 \frac{1}{n} \parallel \mathsf{C}\boldsymbol{\delta} - \mathbf{t} \parallel_2^2 . \tag{8}$$

Note that this holds if higher order derivative are used as a smoothing penalty.

The CV$_{loo}$ score can similarly be approximated by:

$$\tilde{\mathcal{E}}_g^2(\lambda) = \frac{1}{n} \sum_{j=1}^{n} \parallel \mathbf{c}_j^\mathsf{T} \hat{\boldsymbol{\delta}}_{(j)} - \mathbf{s}_j \parallel_2^2 , \tag{9}$$

where $\hat{\boldsymbol{\delta}}_{(j)}$ is the solution of the linear least squares system (8) with all but the j-th correspondence. This way we have approximated the nonlinear least squares problem by a linear least squares one. We call this approximation the GNCV$_{loo}$ score.

Calculating the GNCV$_{loo}$ score using (9) is cheaper than calculating the CV$_{loo}$ score, but it still requires iterating over the n correspondences.

We derive a non-iterative formula that exactly estimate (9). This formula is the following:

$$\tilde{\mathcal{E}}_g^2(\lambda) = \frac{1}{n} \left\| \mathtt{diag}\left(\frac{1}{\mathbf{1} - \mathtt{diag}\,(\hat{\mathsf{C}})} \right) \left(\hat{\mathsf{C}}\mathbf{k} - \mathbf{s} - \mathtt{diag}\,(\mathtt{diag}\,(\hat{\mathsf{C}}))(\mathbf{k} - \mathbf{s}) \right) \right\|_2^2 , \tag{10}$$

where

$$\hat{\mathsf{C}} = \mathsf{C}\mathsf{C}^+, \mathsf{C}^+ \text{ is the pseudoinverse of } \mathsf{C} \text{ and } \mathbf{k} = \frac{(1-\lambda)\mathbf{s} + \lambda\mathbf{t}}{(1-\lambda)^2 + \lambda^2} .$$

$\mathbf{1}$ is a vector of ones and the `diag` operator, similar to the one in Matlab, extracts a diagonal matrix or constructs one from a vector. The derivation of this formula is shown in the Appendix.

Maximizing the GNCV$_{loo}$ score is then done through the global solution of Equation (2) and the closed-form (10): instead of n nonlinear least squares problems only one has to be solved.

As is, this CV method is not robust, in the sense that it does not cope with mismatched correspondences. Therefore, we use RANSAC to robustly estimate as initial solution with $\lambda = 0$ to get $\hat{\mathsf{P}}_\lambda$ in Equation (6), and the dataset is restricted to only correspondences classified as inliers after RANSAC. Moreover, the computation of $\hat{\lambda}$ is carried out by sampling, estimating Equation (2) at steps of 0.01 from 0 to 1. Here 0.01 was experimentally derived as a sufficiently fine discretization step to find the global minimum of the GNCV$_{loo}$ score. Table 1 summarizes the proposed method.

```
Camera Resection Method
    Find an initial robust estimation of $\mathsf{P}$;
    On those points classified as inliers:
        for $\lambda = 0 : 0.01 : 1$
            Find $\hat{\mathsf{P}}_\lambda$, using Gauss-Newton;
            Estimate the GNCV$_{loo}$ score using (10);
        end
    Select $\hat{\mathsf{P}}_\lambda$ with minimum GNCV$_{loo}$ score.
```

Table 1. Automatic camera resection based on our GNCV$_{loo}$ score.

4 Implementation Details

We give some details of our implementation of the reconstruction pipeline. We assume that the camera is calibrated.

The KLT tracker [18] is used to detect and track keypoints in the sequence. Similarly to [19], the first frame is chosen as the first keyframe I_1. I_2 is chosen so that there are as many frames as possible between I_1 and I_2 with at least N feature points in common. Frame I_n is selected as a keyframe if: a) there are as many frames as possible between I_n and I_{n-1}; b) there are at least N point correspondences between I_{n-1} and I_n and c) there are at least M point correspondences between I_{n-2} and I_n. This criterion ensures that there are common matches at least in three consecutive views. In our experiments we used $N = 300$ and $M = 200$.

In order to initialize the 3D reconstruction we use the first image triplet, computing relative camera motion as described in [13]. This process is coupled with RANSAC, in order to have a robust estimate, and the final solution is further refined with bundle adjustment [20]. The resection problem is solved as described in Section 3, using Fiore's linear algorithm [5] to find the initial estimate.

The 3D points are obtained by triangulation considering all image points of the visible tracks up to the current keyframe. A reconstructed point is considered an inlier if a) its computation is well conditioned – we set a threshold on the condition number of the matrix in the linear system that computes the 3D point – and b) if it projects sufficiently close, say by a distance of one pixel, to all associated image points. This requires us to refine the initial estimation of a 3D point based on all observations, including the latest. Therefore, each time a new keyframe is added the tracks visible in it are checked and the list of inliers updated.

For the first 10 views, a full bundle adjustment using all keyframes and all points is performed. After that the computation becomes increasingly expensive, even if the sparseness inherent to the problem is exploited [21]. So we perform local bundle adjustment, i.e. only a subset of keyframe poses are adjusted. Similarly to [14], we choose the last 5 keyframes, while the frames beyond these are locked and not moved. All 3D points visible in the last keyframes are considered, together with all measurements ever made of these points. That is, the reprojection errors are accumulated for the entire tracks backwards in time, regardless of whether the views where the reprojections reside are locked.

5 Experimental Results

In Figure 3 we show the final camera pose obtained by the proposed nonlinear refinement for the cases depicted in Figure 2. We compare the true CV_{loo} score, calculated in the greedy way, with the approximated $GNCV_{loo}$ score proposed in this paper. The optimal values of λ obtained with the CV_{loo} score are respectively 0.05 and 0, while the values obtained with the GNCV_{loo} score are respectively 0.05 and 0.02. It can be seen that the approximation is close to the true score.

We show the effectiveness of our method on synthetic unstable sequences. The dataset consists of 100 points randomly scattered in a sphere of radius 1 meter, centred at the origin. We consider three different scenarios. In the first setting, views are generated by placing cameras along a line in the z-direction, at a distance from the origin of 5.5 up to 7 meters approximately. In the second setting, in order to simulate more unstable cases, the rectilinear trajectory is perturbed along the x-direction, applying Gaussian noise of standard deviation 0.8. In the third setting the trajectory is perturbed in the three directions, with the same noise. In the three cases the number of views is fixed to 10, and Gaussian noise with standard deviation 0.5 is added to the image points.

For each scenario we compare results in terms of distance of the estimated cameras to the ground truth, considering four cases: a) without using nonlinear refinement of camera pose, b) using nonlinear refinement, but without the smoothing penalty, c) with the smoothing term and λ carefully set by hand and kept constant for the whole sequence, and finally d) with λ estimated by CV, as proposed in this paper. For each scenario 50 independent trials are carried out.

Results are shown in Table 2. The distances between the camera centres of the estimated cameras and the ground truth are reported. In the third scenario

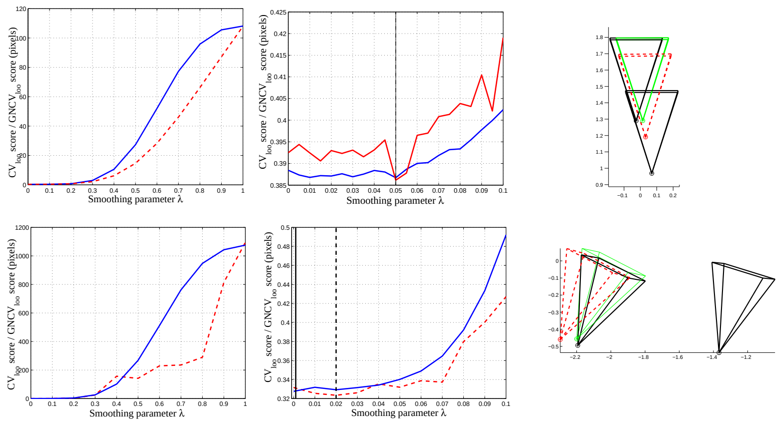


Fig. 3. In the left column, comparison between the true CV$_{loo}$ score (blue thick line) and its approximation (red dashed line) in the examples of Figure 2. The central column shows a zoom around the minimum. The vertical lines indicate the CV$_{loo}$ score minimum (black line) and the GNCV$_{loo}$ score minimum (black dashed line). In the right column, the final pose estimate (green, thin line). In black thick line the ground truth for the previous and current pose, in red dashed line the initial pose estimated by RANSAC.

without nonlinear pose refinement the computation stopped before estimating all cameras in 20% of cases. Adding the smoothing penalty improves stability and accuracy in pose estimation, and our automatic method gives the best results.

	Setting 1				Setting 2				Setting 3			
	a	b	c	d	a	b	c	d	a*	b	c	d
mean	0.1704	0.0916	0.0793	0.0566	0.0766	0.110	0.091	0.044	0.052	0.072	0.090	0.067
min	0.001	0.005	0.004	0.002	0.005	0.004	0.007	0.001	0.003	0.003	0.005	0.004
max	1.018	1.021	0.301	0.383	0.582	0.707	0.614	0.543	0.396	0.479	0.545	0.465

Table 2. Results on synthetic scenes. Mean, minimum and maximum distance of estimated optical centres from the ground truth (in meters) for the three synthetic settings and the cases a) without using nonlinear refinement, b) using nonlinear refinement, but without the smoothing term, c) with the smoothing term and λ carefully set by hand and d) with λ estimated by CV. (*) In this setting 20% of the cases fails.

For real sequences, we first show the results from a video, *Campus*, taken by a calibrated handheld camera (see Figure 4). The trajectory executed is an initial rotation, then a rectilinear part and finally another small rotation, without caring too much about shaking. From 1608 frames of resolution 784×516, without smoothness the reconstruction process stops at the 47th keyframe, as already

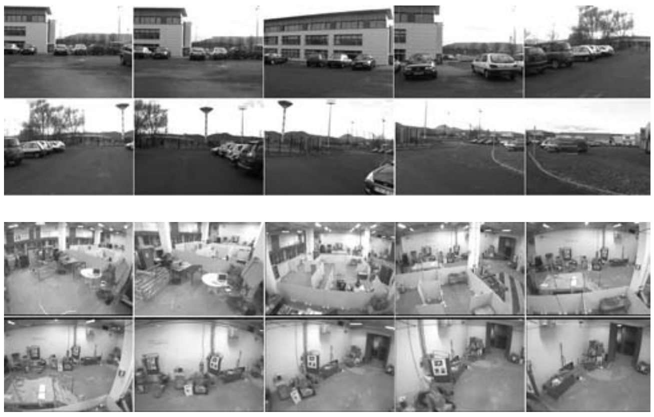


Fig. 4. Thumbnails of the *Campus* (top) and *Laboratory* (bottom) sequences.

displayed in Figure 1. Using the proposed method, instead, all the sequence can be processed, with 135 keyframes extracted, and 5000 points reconstructed with a mean reprojection error of 0.53 pixel. The 3D map produced, with the estimated camera trajectory, can be seen in Figure 5.

The second video, *Laboratory*, is taken with a camera mounted on a Unmanned Autonomous Vehicle (UAV), in an indoor setting. It is made up of 929 frames of resolution 576×784 (see Figure 4). 89 keyframes are calibrated, and the final 3D map is composed of 4421 points (Figure 5) with a mean reprojection error of 0.64 pixel. With no smoothness, the reconstruction process stops in the last rotation.

6 Conclusions

Local camera pose estimation might often be unstable. We proposed adding a smoothing penalty on the camera trajectory and automatically estimating the smoothing parameter, usually manually fixed, using Cross-Validation. The non-iterative closed-form we proposed allows us to solve the problem very efficiently, dropping the complexity one order of magnitude below the straightforward application of Cross-Validation. Experimental results show that the method is effective in improving accuracy and stability in the reconstruction process, preventing several failure cases.

References

1. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. 2nd edn. Cambridge University Press (2003)
2. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07), Nara, Japan (November 2007)

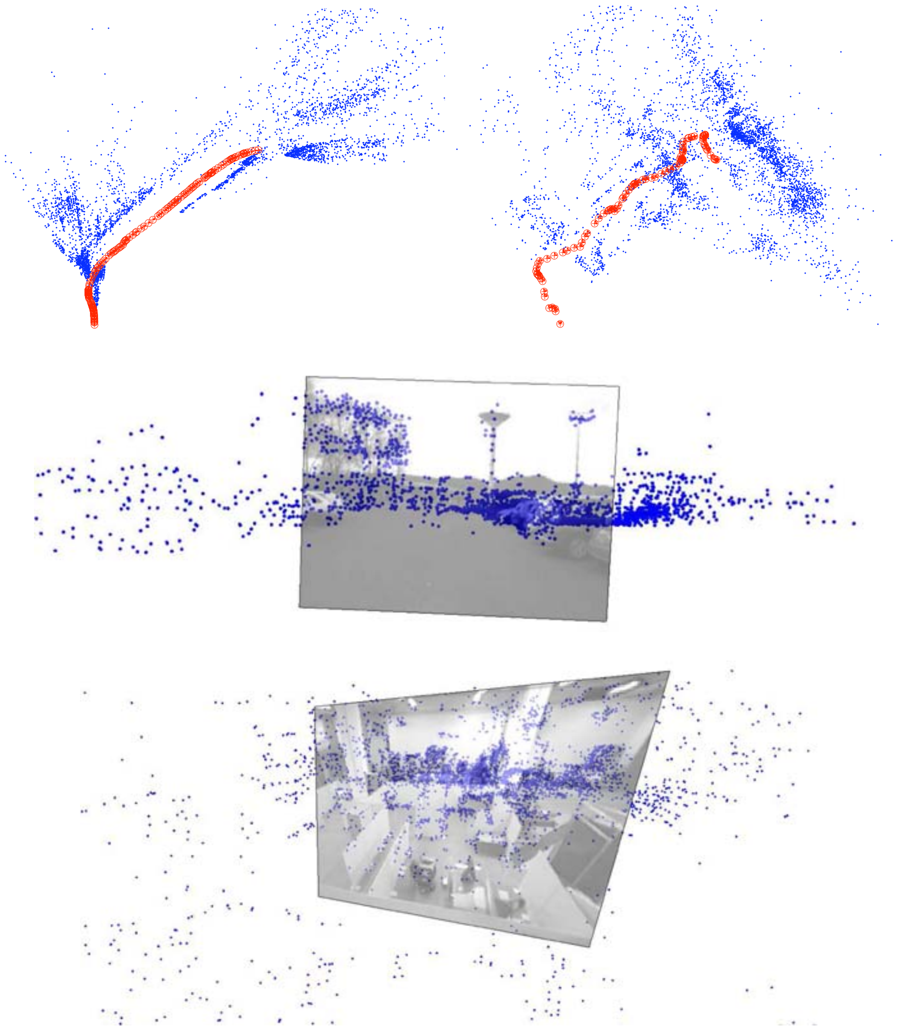


Fig. 5. In the left column, top view of the 3D maps of *Campus* (top) and *Laboratory* (bottom) sequences; in red the estimated cameras. The right column show a perspective view of respectively *Campus* and *Laboratory* maps, where for visualisation understanding one keyframe of the sequence is superimposed.

3. Pollefeys, M., Gool, L.V., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., Koch, R.: Visual modeling with a hand-held camera. International Journal of Computer Vision **59**(3) (2004) 207–232
4. Clipp, B., Welch, G., Frahm, J.M., Pollefeys, M.: Structure from motion via a two-stage pipeline of extended kalman filters. In: British Machine Vision Conference. (2007)

5. Fiore, P.D.: Efficient linear solution of exterior orientation. IEEE Transactions on Pattern Analysis and Machine Intelligence **23**(2) (2001) 140–148
6. Haralick, R., Lee, C., Ottenberg, K., Nolle, M.: Review and analysis of solutions of the three point perspective pose estimation problem. International Journal of Computer Vision **13**(3) (1994) 331–356
7. Fischler, M.A., Bolles, R.C.: Random Sample Consensus: a paradigm model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6) (June 1981) 381–395
8. Gentle, J.E., Hardle, W., Mori, Y.: Handbook of Computational Statistics. Springer-Verlag (2004)
9. Eade, E., Drummond, T.: Scalable monocular vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2006) 469–476
10. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: Monoslam: Real-time single camera slam. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(6) (June 2007) 1052–1067
11. Torr, P.H.S.: Bayesian model estimation and selection for epipolar geometry and generic manifold fitting. International Journal of Computer Vision **50**(1) (2002) 35–61
12. Thomahlen, T., Broszio, H., Weissenfeld, A.: Keyframe selection for camera motion and structure estimation from multiple views. In: Proceedings of the European Conference on Computer Vision. (2004) 523
13. Nistér, D., Naroditsky, O., Bergen, J.: Visual odometry. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Volume 2. (2004) 652–659
14. Hengels, C., Stewénius, H., Nistér, D.: Bundle adjustment rules. In: Photogrammetric Computer Vision. (September 2006)
15. Olsen, S.I., Bartoli, A.: Using priors for improving generalization in non-rigid structure-from-motion. In: British Machine Vision Conference. (2007)
16. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press (1995)
17. Wahba, G., Wold, S.: A completely automatic french curve: Fitting spline functions by Cross-Validation. Communications in Statistics **4** (1975) 1–17
18. Shi, J., Tomasi, C.: Good features to track. Technical Report 93-1399, Department of Computer Science, Cornell University, Ithaca, NY 14853-7501 (November 1993)
19. Royer, E., Lhuillier, M., Dhome, M., Lavest, J.: Monocular vision for mobile robot localization and autonomous navogation. International Journal of Computer Vision **74**(3) (2007) 237–260
20. Lourakis, M., Argyros, A.: The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece (Aug. 2004) Available from `http://www.ics.forth.gr/~lourakis/sba`.
21. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle adjustment - a modern synthesis. In: ICCV '99: Proceedings of the International Workshop on Vision Algorithms. (2000) 298–372

A Deriving a Non Iterative Formula for GNCV_{loo}

We show that there exists a non-iterative formula that exactly estimate (9).

First, given the smoothing parameter λ, $\hat{\boldsymbol{\delta}}$ in Equation (8) is solved through:

$$\begin{aligned}
\hat{\boldsymbol{\delta}} &= \arg\min_{\boldsymbol{\delta}} \tilde{\mathcal{E}}(\boldsymbol{\delta}, \lambda) \\
&= \arg\min_{\boldsymbol{\delta}} \left\| \begin{bmatrix} (1-\lambda)\mathsf{C} \\ \lambda\mathsf{C} \end{bmatrix} \boldsymbol{\delta} - \begin{bmatrix} (1-\lambda)\mathbf{s} \\ \lambda\mathbf{t} \end{bmatrix} \right\|_2^2 \\
&= ((1-\lambda)^2 \mathsf{C}^\mathsf{T}\mathsf{C} + \lambda^2 \mathsf{C}^\mathsf{T}\mathsf{C})^{-1} \mathsf{C}^\mathsf{T}((1-\lambda)^2\mathbf{s} + \lambda^2\mathbf{t}) \\
&= \frac{1}{(1-\lambda)^2 + \lambda^2} (\mathsf{C}^\mathsf{T}\mathsf{C})^{-1}\mathsf{C}^\mathsf{T}((1-\lambda)^2\mathbf{s} + \lambda^2\mathbf{t}) \\
&= \frac{1}{\|\boldsymbol{\omega}\|_2^2} \mathsf{C}^+ \mathsf{R} \boldsymbol{\omega}_s \ , \qquad (11)
\end{aligned}$$

where C^+ is the pseudo-inverse of C, $\boldsymbol{\omega}^\mathsf{T} = [(1-\lambda) \ \ \lambda]$, $\boldsymbol{\omega}_s^\mathsf{T} = [(1-\lambda)^2 \ \ \lambda^2]$ and $\mathsf{R} = [\mathbf{s} \ \ \mathbf{t}]$.

We define $\mathbf{e}_j$ as a zero vector with one at the j-th element and $\mathsf{K}_j = \mathrm{I} - \mathtt{diag}(\mathbf{e}_j)$. We recall that $\mathsf{K}_j\mathsf{K}_j = \mathsf{K}_j$ and $\mathsf{K}_j^\mathsf{T} = \mathsf{K}_j$. Then $\hat{\boldsymbol{\delta}}_{(j)}$ is solved through:

$$\begin{aligned}
\hat{\boldsymbol{\delta}}_{(j)} &= \arg\min_{\boldsymbol{\delta}} \left\| \begin{bmatrix} (1-\lambda)\mathsf{K}_j\mathsf{C} \\ \lambda\mathsf{K}_\mathsf{j}\mathsf{C} \end{bmatrix} \boldsymbol{\delta} - \begin{bmatrix} (1-\lambda)\mathsf{K}_j\mathbf{s} \\ \lambda\mathsf{K}_j\mathbf{t} \end{bmatrix} \right\|_2^2 \\
&= \frac{1}{\|\boldsymbol{\omega}\|_2^2} (\mathsf{K}_j\mathsf{C})^+ \mathsf{K}_j \mathsf{R} \boldsymbol{\omega}_s \ . \qquad (12)
\end{aligned}$$

$\hat{\boldsymbol{\delta}}_{(j)}$ can be expressed alternatively as this following Lemma says:

Lemma 1. $\hat{\boldsymbol{\delta}}_{(j)}$ *as defined by Equation (12) is given by the following equation:*

$$\hat{\boldsymbol{\delta}}_{(j)} = \frac{1}{\|\boldsymbol{\omega}\|_2^2} \mathsf{C}^+ \tilde{\mathsf{R}}_j \boldsymbol{\omega}_s, \ \ \textit{with} \ \ \tilde{\mathsf{R}}_j = \mathsf{K}_j\mathsf{R} + \frac{\|\boldsymbol{\omega}\|_2^2}{\|\boldsymbol{\omega}_s\|_2^2} (\mathrm{I} - \mathsf{K}_j)\mathsf{C}\hat{\boldsymbol{\delta}}_{(j)}\boldsymbol{\omega}_s^\mathsf{T} \ . \qquad (13)$$

Proof. We start by expanding the right-hand side of Equation (13):

$$\frac{1}{\|\boldsymbol{\omega}\|_2^2} \mathsf{C}^+ \tilde{\mathsf{R}}_j \boldsymbol{\omega}_s = \frac{1}{\|\boldsymbol{\omega}\|_2^2} \left(\mathsf{C}^+\mathsf{K}_j\mathsf{R}\boldsymbol{\omega}_s + \frac{\|\boldsymbol{\omega}\|_2^2}{\|\boldsymbol{\omega}_s\|_2^2} \mathsf{C}^+(\mathrm{I} - \mathsf{K}_j)\mathsf{C}\hat{\boldsymbol{\delta}}_{(j)}\boldsymbol{\omega}_s^\mathsf{T}\boldsymbol{\omega}_s \right),$$

and as $\boldsymbol{\omega}_s^\mathsf{T}\boldsymbol{\omega}_s = \|\boldsymbol{\omega}_s\|_2^2$ we can simplify:

$$\frac{1}{\|\boldsymbol{\omega}\|_2^2} \mathsf{C}^+ \tilde{\mathsf{R}}_j \boldsymbol{\omega}_s = \frac{1}{\|\boldsymbol{\omega}\|_2^2} (\mathsf{C}^+\mathsf{K}_j\mathsf{R}\boldsymbol{\omega}_s + \|\boldsymbol{\omega}\|_2^2 \mathsf{C}^+\mathsf{C}\hat{\boldsymbol{\delta}}_{(j)} - \|\boldsymbol{\omega}\|_2^2 \mathsf{C}^+\mathsf{K}_j\mathsf{C}\hat{\boldsymbol{\delta}}_{(j)}) \ .$$

The second term reduces to $\hat{\boldsymbol{\delta}}_{(j)}$ since $\mathsf{C}^+\mathsf{C} = \mathrm{I}$. The third term, replacing $\hat{\boldsymbol{\delta}}_{(j)}$ with its expression (12), expands as:

$$\begin{aligned}
\|\boldsymbol{\omega}\|_2^2 \mathsf{C}^+\mathsf{K}_j\mathsf{C}\hat{\boldsymbol{\delta}}_{(j)} &= \mathsf{C}^+\mathsf{K}_j\mathsf{C} \ (\mathsf{K}_j\mathsf{C})^+\mathsf{K}_j\mathsf{R}\boldsymbol{\omega}_s \\
&= (\mathsf{C}^\mathsf{T}\mathsf{C})^{-1} \underbrace{\mathsf{C}^\mathsf{T}\mathsf{K}_j\mathsf{C}(\mathsf{C}^\mathsf{T}\mathsf{K}_j\mathsf{C})^{-1}}_{\mathsf{I}} \mathsf{C}^\mathsf{T}\mathsf{K}_j\mathsf{R}\boldsymbol{\omega}_s \\
&= \mathsf{C}^+\mathsf{K}_\mathsf{j}\mathsf{R}\boldsymbol{\omega}_s \ ,
\end{aligned}$$

and the overall expression simplifies to:

$$\frac{1}{\|\boldsymbol{\omega}\|_2^2}\mathsf{C}^+\tilde{\mathsf{R}}_j\boldsymbol{\omega}_s = \hat{\boldsymbol{\delta}}_{(j)} \ . \qquad \square$$

The projection of the j-th data with the global model $\hat{\boldsymbol{\delta}}$ is $\mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}}$ and with the partial model $\hat{\boldsymbol{\delta}}_{(j)}$ is $\mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}}_{(j)}$. Using Equations (11) and (12) we can rewrite these projections as:

$$\mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}} = \frac{1}{\|\boldsymbol{\omega}\|_2^2}\mathbf{c}_j^\mathsf{T}\mathsf{C}^+\mathsf{R}\boldsymbol{\omega}_s \ = \ \frac{1}{\|\boldsymbol{\omega}\|_2^2}\hat{\mathbf{c}}_j^\mathsf{T}\mathsf{R}\boldsymbol{\omega}_s \ , \tag{14}$$

$$\mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}}_{(j)} = \frac{1}{\|\boldsymbol{\omega}\|_2^2}\mathbf{c}_j^\mathsf{T}\mathsf{C}^+\tilde{\mathsf{R}}_j\boldsymbol{\omega}_s = \frac{1}{\|\boldsymbol{\omega}\|_2^2}\hat{\mathbf{c}}_j^\mathsf{T}\tilde{\mathsf{R}}_j\boldsymbol{\omega}_s \ , \tag{15}$$

with $\hat{\mathbf{c}}_j^\mathsf{T}$ the j-th row of the hat matrix $\hat{\mathsf{C}} = \mathsf{C}\mathsf{C}^+$.

We note that $(\mathsf{I}-\mathsf{K}_j)\mathsf{C} = \mathbf{e}_j\mathbf{c}_j^\mathsf{T}$. Taking the difference between the two predictions and factoring we obtain:

$$\begin{aligned}
\mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}} - \mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}}_{(j)} &= \frac{1}{\|\boldsymbol{\omega}\|_2^2}\hat{\mathbf{c}}_j^\mathsf{T}(\mathsf{R}-\tilde{\mathsf{R}}_j)\,\boldsymbol{\omega}_s \\
&= \frac{1}{\|\boldsymbol{\omega}\|_2^2}\hat{\mathbf{c}}_j^\mathsf{T}\left(\mathsf{R}-\mathsf{K}_j\mathsf{R}-\frac{\|\boldsymbol{\omega}\|_2^2}{\|\boldsymbol{\omega}_s\|_2^2}\,\mathbf{e}_j\mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}}_{(j)}\boldsymbol{\omega}_s^\mathsf{T}\right)\,\boldsymbol{\omega}_s \\
&= \frac{1}{\|\boldsymbol{\omega}\|_2^2}\hat{\mathbf{c}}_j^\mathsf{T}(\mathbf{e}_j\mathbf{r}_j^\mathsf{T}\boldsymbol{\omega}_s - \|\boldsymbol{\omega}\|_2^2\,\mathbf{e}_j\mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}}_{(j)}) \\
&= \frac{1}{\|\boldsymbol{\omega}\|_2^2}\hat{c}_{jj}(\mathbf{r}_j^\mathsf{T}\boldsymbol{\omega}_s - \|\boldsymbol{\omega}\|_2^2\,\mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}}_{(j)}) \ ,
\end{aligned}$$

where $\hat{c}_{jj}$ is the j-th diagonal element of the matrix $\hat{\mathsf{C}}$. Rearranging the terms gives:

$$\mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}} - \frac{1}{\|\boldsymbol{\omega}\|_2^2}\,\hat{c}_{jj}\mathbf{r}_j^\mathsf{T}\boldsymbol{\omega}_s = (1-\hat{c}_{jj})\mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}}_{(j)} \ .$$

Subtracting $(1-\hat{c}_{jj})\mathbf{s}_j$ on both sides we have:

$$\mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}} - \frac{1}{\|\boldsymbol{\omega}\|_2^2}\,\hat{c}_{jj}\mathbf{r}_j^\mathsf{T}\boldsymbol{\omega}_s - \mathbf{s}_j + \hat{c}_{jj}\mathbf{s}_j = (1-\hat{c}_{jj})(\mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}}_{(j)} - \mathbf{s}_j) \ ,$$

from which:

$$\mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}}_{(j)} - \mathbf{s}_j = \frac{1}{1-\hat{c}_{jj}}\left(\mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}} - \mathbf{s}_j + \hat{c}_{jj}\left(\mathbf{s}_j - \frac{1}{\|\boldsymbol{\omega}\|_2^2}\,\mathbf{r}_j^\mathsf{T}\boldsymbol{\omega}_s\right)\right) .$$

We observe that $\mathbf{c}_j^\mathsf{T}\hat{\boldsymbol{\delta}}_{(j)} - \mathbf{s}_j$ is the residual vector of the j-th measurement, as in Equation (9). Replacing $\hat{\boldsymbol{\delta}}$ from Equation (11) and summing the squared norm over j we get, after some rewriting, the non-iterative formula (10):

$$\tilde{\mathcal{E}}_g^2(\lambda) = \frac{1}{n}\left\|\,\mathtt{diag}\left(\frac{1}{\mathbf{1}-\mathtt{diag}\,(\hat{\mathsf{C}})}\right)\left(\hat{\mathsf{C}}\mathbf{k} - \mathbf{s} - \mathtt{diag}\,(\mathtt{diag}\,(\hat{\mathsf{C}}))(\mathbf{k}-\mathbf{s})\right)\right\|_2^2 .$$