MODULAR BUNDLE ADJUSTMENT FOR PHOTOGRAMMETRIC COMPUTATIONS

Niclas Börlin, Arnadi Murtiyoso, Pierre Grussenmeyer, Fabio Menna, Erica Nocerino

MODULAR BUNDLE ADJUSTMENT FOR PHOTOGRAMMETRIC COMPUTATIONS

N. Börlin[1,*], A. Murtiyoso[2], P. Grussenmeyer[2], F. Menna[3], E. Nocerino[3]

[1] Department of Computing Science, Umeå University, Sweden — niclas.borlin@cs.umu.se
[2] Photogrammetry and Geomatics Group, ICube Laboratory UMR 7357, INSA Strasbourg, France — (arnadi.murtiyoso, pierre.grussenmeyer)@insa-strasbourg.fr
[3] 3D Optical Metrology (3DOM) unit, Bruno Kessler Foundation (FBK), Trento, Italy - (fmenna, nocerino)@fbk.eu

Commission II, WG II/1

KEY WORDS: Bundle adjustment, Camera model, Analytical Jacobians, Software, Photogrammetry, Tilt-shift lens

ABSTRACT:

In this paper we investigate how the residuals in bundle adjustment can be split into a composition of simple functions. According to the chain rule, the Jacobian (linearisation) of the residual can be formed as a product of the Jacobians of the individual steps. When implemented, this enables a modularisation of the computation of the bundle adjustment residuals and Jacobians where each component has limited responsibility. This enables simple replacement of components to e.g. implement different projection or rotation models by exchanging a module.
The technique has previously been used to implement bundle adjustment in the open-source package DBAT (Börlin and Grussenmeyer, 2013) based on the Photogrammetric and Computer Vision interpretations of Brown (1971) lens distortion model. In this paper, we applied the technique to investigate how affine distortions can be used to model the projection of a tilt-shift lens. Two extended distortion models were implemented to test the hypothesis that the ordering of the affine and lens distortion steps can be changed to reduce the size of the residuals of a tilt-shift lens calibration.
Results on synthetic data confirm that the ordering of the affine and lens distortion steps matter and is detectable by DBAT. However, when applied to a real camera calibration data set of a tilt-shift lens, no difference between the extended models was seen. This suggests that the tested hypothesis is false and that other effects need to be modelled to better explain the projection. The relatively low implementation effort that was needed to generate the models suggest that the technique can be used to investigate other novel projection models in photogrammetry, including modelling changes in the 3D geometry to better understand the tilt-shift lens.

1. INTRODUCTION

The mathematical problem that is solved by the Bundle adjustment (BA) process includes a residual between two components. One residual component is the simulated projection of an object point according to the camera model at hand. The other component is computed from the corresponding image measurement[1]. As a typical presentation, consider equation (1) below (Luhmann et al., 2014, equation (4.94)) Ignoring $\Delta x'$ and $\Delta y'$, equation (1) completely describes the projection of the object point (OP) (X, Y, Z) through a pinhole camera with principal point (x'_0, y'_0) and camera constant z' placed at (X_0, Y_0, Z_0) and with an orientation given by the matrix R.

*Corresponding author

[1] For simplicity, in this paper we ignore additional observations of object point coordinates, etc.

Another requirement of the bundle adjustment is the linearisation of equation (1) with respect to any parameter that is to be estimated (see e.g. Kraus (1993, Sec. 5.3.2), Wolf and Dewitt (2000, App. D-4), Mikhail et al. (2001, App. C.3), or Luhmann et al. (2014, Sec. 4.4.2)). Some partial derivatives of equation (1) are given in equation (2) below (Luhmann et al., 2014, equation (4.96)). In equation (2), k_X and k_Y are the respective numerators of equation (1), and N is the denominator.

What may not be immediately obvious from equation (1) is that the computation can be split into a sequence of basic operations. The goal of this paper is to show how that can be done and that, if each operation is treated as a module with a responsibility for computing both the result of the operation and its linearisation (Jacobians) with respect to any parameter, the computation of the analytical Jacobians can be greatly simplified.

$$\begin{aligned} x' &= x'_0 + z' \frac{r_{11}(X - X_0) + r_{21}(Y - Y_0) + r_{31}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)} + \Delta x', \\ y' &= y'_0 + z' \frac{r_{12}(X - X_0) + r_{22}(Y - Y_0) + r_{32}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)} + \Delta y'. \end{aligned} \tag{1}$$

$$\begin{aligned} \frac{\partial x'}{\partial X} &= -\frac{z'}{N^2}(r_{13}k_X - r_{11}N), & \frac{\partial x'}{\partial Y} &= -\frac{z'}{N^2}(r_{23}k_X - r_{21}N), & \frac{\partial x'}{\partial Z} &= -\frac{z'}{N^2}(r_{33}k_X - r_{31}N), \\ \frac{\partial y'}{\partial X} &= -\frac{z'}{N^2}(r_{13}k_Y - r_{12}N), & \frac{\partial y'}{\partial Y} &= -\frac{z'}{N^2}(r_{23}k_Y - r_{22}N), & \frac{\partial y'}{\partial Z} &= -\frac{z'}{N^2}(r_{33}k_Y - r_{32}N), \end{aligned} \tag{2}$$

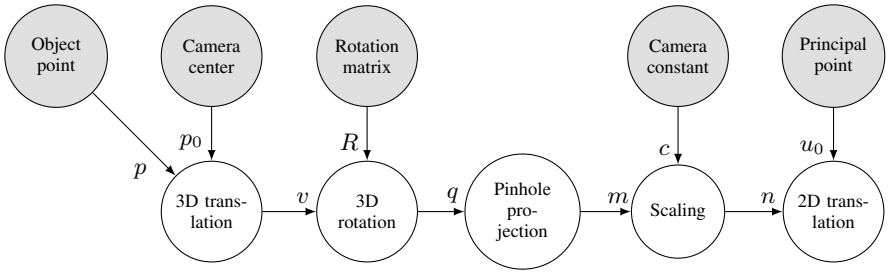

Figure 1: The computational chain corresponding to equation (5). The gray circles indicate bundle adjustment parameters. The white circles indicate component functions in equation (4). The arrows indicate how the parameters and results are propagated. The arrow labels indicate the name of the formal (input) parameter of the corresponding function.

2. SPLITTING THE RAYS

2.1 Basic functions

If we group the parameters as

$$\begin{aligned} p &= \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \quad p_0 = \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix}, \quad q = \begin{pmatrix} k_X \\ k_Y \\ N \end{pmatrix}, \\ u &= \begin{pmatrix} x' \\ y' \end{pmatrix}, \quad u_0 = \begin{pmatrix} x'_0 \\ y'_0 \end{pmatrix}, \quad \Delta u = \begin{pmatrix} \Delta x' \\ \Delta y' \end{pmatrix}, \end{aligned} \tag{3}$$

and introduce the basic functions

$$T_3(p, p_0) = p + p_0, \qquad \text{3D translation} \tag{4a}$$
$$L(R, v) = Rv, \qquad \text{3D rotation} \tag{4b}$$
$$H(q) = \frac{1}{q_3} \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}, \qquad \text{Pinhole projection} \tag{4c}$$
$$S(c, m) = cm, \qquad \text{Scaling} \tag{4d}$$
$$T_2(n, u_0) = n + u_0, \qquad \text{2D translation,} \tag{4e}$$

we may write equation (1) as the following composition:

$$\begin{aligned} u &= g(p, p_0, R, z', u_0) \\ &\equiv T_2(S(z', H(L(R^T, T_3(p, -p_0)))), u_0). \end{aligned} \tag{5}$$

In the definition of the functions in equation (4), generality has been favoured over typical usage to simplify future extensions. For instance, the function $L(R, v)$ describes a 3D linear transformation whereas the typical usage is with a rotation matrix R. Similarly, the translation $T_3(p, p_0)$ is defined with a positive sign on p_0 whereas the typical usage would be $T_3(p, -p_0)$.

The algorithm corresponding to equation (5) is shown in Algorithm 1 with a graphical representation in Figure 1.

2.2 Linearisation

The linearisation of composed functions are governed by the chain rule (see Appendix A). For example, the Jacobian of the projection function g in equation (5) with respect to the object point coordinates p is the matrix product of five Jacobians

$$\begin{aligned} \left[\frac{dg}{dp}\right] = &\left[\frac{dT_2}{dn}\right]_{n=S(z', H(L(R^T, T_3(p, -p_0))))} \\ &\cdot \left[\frac{dS}{dm}\right]_{m=H(L(R^T, T_3(p, -p_0)))} \\ &\cdot \left[\frac{dH}{dq}\right]_{q=L(R^T, T_3(p, -p_0))} \\ &\cdot \left[\frac{dL}{dv}\right]_{v=T_3(p, -p_0)} \\ &\cdot \left[\frac{dT_3}{dp}\right], \end{aligned} \tag{6}$$

where $\cdot$ indicates matrix multiplication and the subscripts indicate at which point the respective Jacobians are evaluated. A detailed inspection of equation (6) reveals that it is identical to equation (2).

To better highlight the chaining, a simplified notation that leaves the evaluation point implicit may be used:

$$\left[\frac{dg}{dp}\right] = \left[\frac{dT_2}{dS}\right] \left[\frac{dS}{dH}\right] \left[\frac{dH}{dL}\right] \left[\frac{dL}{dT_3}\right] \left[\frac{dT_3}{dp}\right]. \tag{7}$$

2.3 Modularisation

Algorithm 2 is an extension to Algorithm 1 to also compute the Jacobian according to Equation (6). In Algorithm 2, each function is assumed to compute both the function value proper and the Jacobians with respect to each parameter. From Algorithm 2, we see that each function will indeed have all the necessary information to compute the function value and its Jacobians at the proper

Algorithm 1 Pinhole projection corresponding to equation (5).

procedure PINHOLENOJAC(p, p_0, R, c, u_0)
  $a_1 \leftarrow T_3(p, -p_0)$
  $a_2 \leftarrow L(R^T, a_1)$
  $a_3 \leftarrow H(a_2)$
  $a_4 \leftarrow S(c, a_3)$
  $a_5 \leftarrow T_2(a_4, u_0)$
  return a_5
end procedure

Algorithm 2 Algorithm 1 extended to compute Jacobians. Each function is capable of computing the Jacobians with respect to each of its parameters. This example only shows the Jacobians necessary to compute $\left[\frac{dg}{dp}\right]$ according to equation (6).

procedure PINHOLEWITHJAC(p, p_0, R, c, u_0)
  $(a_1, J_p) \leftarrow T_3(p, -p_0)$ ▷ $J_p = \left[\frac{dT_3}{dp}\right]$
  $(a_2, J_v) \leftarrow L(R^T, a_1)$ ▷ $J_v = \left[\frac{dL}{dv}\right]_{v=a_1}$
  $(a_3, J_q) \leftarrow H(a_2)$ ▷ $J_q = \left[\frac{dH}{dq}\right]_{q=a_2}$
  $(a_4, J_m) \leftarrow S(c, a_3)$ ▷ $J_m = \left[\frac{dS}{dm}\right]_{m=a_3}$
  $(a_5, J_n) \leftarrow T_2(a_4, u_0)$ ▷ $J_n = \left[\frac{dT_2}{dn}\right]_{n=a_4}$
  $J \leftarrow J_n J_m J_q J_v J_p$ ▷ $J = \left[\frac{dg}{dp}\right]$
  return (a_5, J)
end procedure

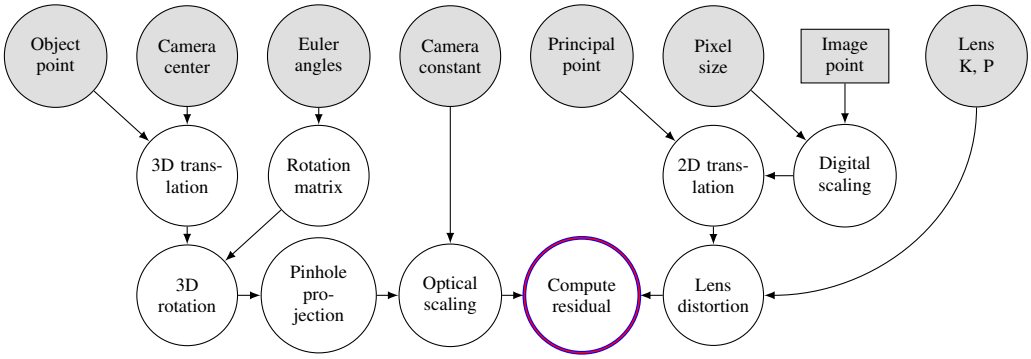

Figure 2: The computational chain implemented in DBAT (Börlin and Grussenmeyer, 2013). In the photogrammetric formulation, the optical scaling in the camera results in image coordinates expressed in physical units, e.g. mm. The image coordinates are scaled from pixels to mm before the Brown (1971) polynomials are used to ”correct” the measured image coordinates for lens distortion. The residual (thick circle) is computed as the difference between the projected ideal point and the corrected point.

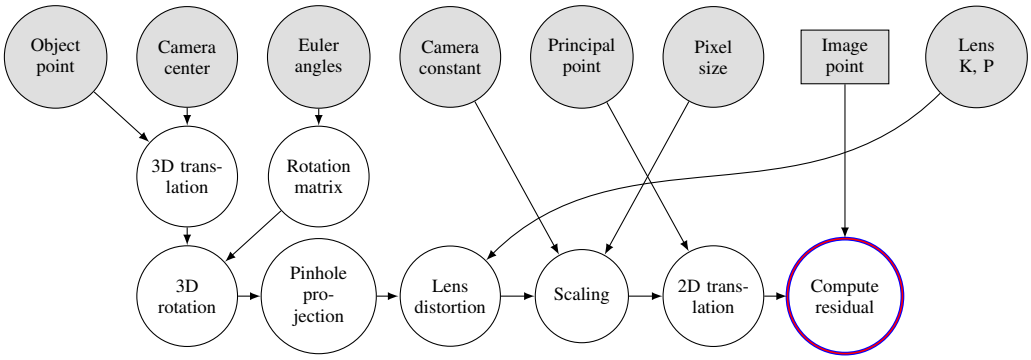

Figure 3: The DBAT implementation of the Computer Vision formulation of the Brown (1971) lens distortion model (Tsai, 1987; Heikkilä and Silvén, 1997; Zhang, 2000). The Brown polynomials are used to ”add” lens distortion to the projected ideal point in normalised units before the coordinates are scaled directly to pixels. The residual (thick circle) is computed between the measured point and the ideal projected point with added lens distortion. The same modules have been used as in Figure 2.

evaluation point. Thus, by adding the requirement that each function should be able to compute its own Jacobians, we may modularise the residual computations and use the functions as building blocks. Furthermore, as each module is self-contained, it is possible to validate the analytical Jacobian of each module independently.

2.4 Photogrammetric modules

In addition to the functions listed in equation (4), we add the computation of the rotation matrix R from the $\omega - \phi - \kappa$ Euler angles (Förstner and Wrobel, 2004, eqs. (2.128)–(2.130))

$$R(\omega, \phi, \kappa) = R_3(\kappa)R_2(\phi)R_1(\omega), \tag{8a}$$

the Brown (1971) lens distortion model

$$D(u, K, P) = u + d_r(u, K) + d_t(u, P), \tag{8b}$$

and the 2D affine transformation

$$A_2(u, b) = \begin{pmatrix} 1 + b_1 & b_2 \\ 0 & 1 \end{pmatrix} u. \tag{8c}$$

In equation (8b), the vectors K and P contain the radial and tangential distortion coefficients, respectively. In equation (8c), the scalar b_1 controls the aspect ratio and b_2 controls the skew. For more details and derivation of the Jacobians, see Appendix B.

3. USAGE

The modular technique has previously been used in the open-source Damped Bundle Adjustment Toolbox (DBAT)[2] package (Börlin and Grussenmeyer, 2013, 2016). In the 2016 paper, the technique was used to implement two bundle pipelines that used different adaptations of the Brown (1971) lens distortion model. In the Photogrammetric formulation, the Brown polynomials are used to ”correct” for the effect of lens distortion on the measured image coordinates. In contrast, the formulation largely adopted by the Computer Vision community uses the same polynomials to ”add” lens distortion to the ideal projection of object points (Tsai, 1987; Heikkilä and Silvén, 1997; Zhang, 2000). The two pipelines are contrasted in figures 2 and 3.

In this paper, we have used the modular technique to investigate the effect of the relative ordering of an affine transformation and lens distortion on the calibration of a tilt-shift lens. Two extensions of the photogrammetric pipeline of Figure 2 was implemented in DBAT. The reference implementation (Model 2) has no affinity. In Model 3, the affinity was applied *before* lens distortion. In Model 4, the affinity was applied *after*. The corresponding functions are:

$$r_2 = D(T_2(S(s, u), -u_0), K, P), \tag{9a}$$
$$r_3 = D(A_2(T_2(S(s, u), -u_0), b), K, P), \tag{9b}$$
$$r_4 = A_2(D(T_2(S(s, u), -u_0), K, P), b), \tag{9c}$$

[2] `https://github.com/niclasborlin/dbat`

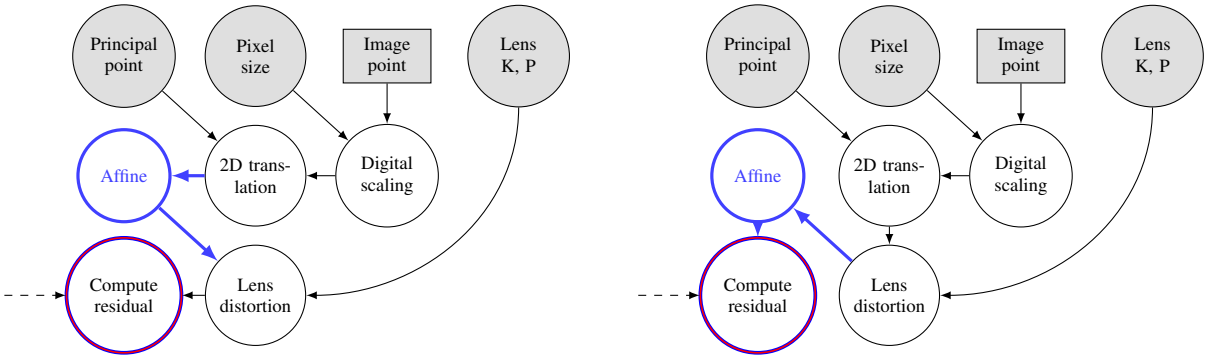

Figure 4: Two versions of the right-hand side of Figure 2 with different relative placement of the affine transformation with respect to lens distortion. In Model 3 (left), the affine distortion was applied before lens distortion. In Model 4 (right), the affine distortion was applied after lens distortion. Note that in both cases, the digital scaling uses square pixel sizes. Any non-unit aspect ratio is handled by the affine step.

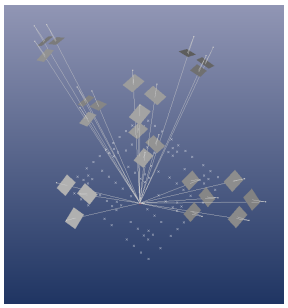

Figure 5: The synthetic network consisted of 24 cameras at varying roll angles. The simulated targets were in a 3D configuration with a largest dimension of 1000 mm.

where s is the pixel size and u the measured image coordinates. The models are illustrated in Figure 4. As an illustration of the (low) level of complexity needed, the difference in the computation of the Jacobian with respect to the principal point u_0 was limited to a inserting a Jacobian at the proper place in the matrix product (in addition to a change in the evaluation points):

$$\left[\frac{dr_2}{du_0}\right] = -\left[\frac{dD}{du}\right]\left[\frac{dT_2}{du_0}\right], \tag{10a}$$

$$\left[\frac{dr_3}{du_0}\right] = -\left[\frac{dD}{du}\right]\left[\frac{dA_2}{du}\right]\left[\frac{dT_2}{du_0}\right], \tag{10b}$$

$$\left[\frac{dr_4}{du_0}\right] = -\left[\frac{dA_2}{du}\right]\left[\frac{dD}{du}\right]\left[\frac{dT_2}{du_0}\right]. \tag{10c}$$

4. EXPERIMENTS AND RESULTS

4.1 Simulation experiment

The first experiment was constructed to investigate the effect, if any, the difference in affine-lens distortion ordering had. Two sets of synthetic data were generated, where simulated error-free image observations were generated by back-projection of known 3D object coordinates and with known exterior orientation parameters (EO), and camera calibration parameters (IO) of a strong self-calibration network. The network consisted of 24 cameras at varying roll angles. About 100 targets were simulated in a 3D configuration with a largest dimension of 1000 mm (Figure 5).

The following algorithms were used to simulate Model 3 and Model 4 (note that the order is reversed compared to Figure 4 as we are building image observations):

Figure 6: The Nikon D750 DSLR camera with the PC-E Micro NIKKOR 45 mm f/2.8D ED tilt-shift lens used in this paper.

Model 3
1. Collinearity equations (3D translation, 3D rotation, pinhole projection and optical scaling).
2. Introduce lens distortion (iterative, [mm]).
3. Convert from mm to pixels using a non-square pixel size corresponding to $b_1 = 0.01218$.
4. Introduce the principal point [pixel].

Model 4
1. Collinearity equations (3D translation, 3D rotation, pinhole projection and optical scaling).
2. Apply an affine transformation of the image coordinates corresponding to $b_1 = 0.01218$.
3. Introduce lens distortion (iterative, [mm]).
4. Convert from mm to pixels using a square pixel size.
5. Introduce the principal point [pixel].

Both simulations used a skew (shear) parameter of $b_2 = 0$. The algorithms were implemented in software developed in-house at FBK and not by DBAT.

Each synthetic data set was analysed by a self-calibration bundle adjustment using DBAT models 2, 3, and 4. The datum problem was solved by fixing the EO parameters of one camera and one coordinate of another. No control points were used and the prior weight for the image observations corresponded to a sigma of 0.1 pixels. The following parameters were estimated; the focal length, the principal point, the radial distortion parameters K_1-K_3, and the tangential distortion parameters P_1-P_2. For models 3 and 4, the affine parameters b_1-b_2 were also estimated. The quality of each analysis was evaluated in image space (σ_0 and 2D image point RMS) and object space (3D RMSE between the true and estimated OP coordinates). The results are given in Table 1. When the correct estimation model was used, the simulated b_1 value was recovered to the number of available decimals and the internal and external residuals were effectively zero. When the wrong model was used, the residuals were significantly higher.

Table 1: Assessment of the analysis of the synthetic data sets. The IO parameters columns indicate what IO parameters and how many (n) were included in the estimation. The point RMS is the average residual over all image observations. The 3D RMSE is the average error over all object points. The $\hat{b}_1$ column shows the estimated b_1 value.

Simulated	DBAT	IO parameters			Internal assessment		External assessment
dataset	model	parameters	n	$\hat{b}_1$	σ_0	Point RMS (pixels)	3D RMSE (μm)
3	2	P	8	0	34.2847	4.56	620.60
	3	b_1 b_2 P	10	0.01218	0.0003	0.00	0.01
	4	b_1 b_2 P	10	0.01223	0.3773	0.05	8.26
4	2	P	8	0	34.1330	4.54	616.59
	3	b_1 b_2 P	10	0.01212	0.3697	0.05	8.33
	4	b_1 b_2 P	10	0.01218	0.0003	0.00	0.01

Table 2: Assessment of DBAT estimation models 2, 3, and 4 on the real-world data set. The IO parameters and assessment are as in Table 1, except that the target coordinates computed from the NORMAL data set was used as the true data. The estimated b_1 value was about 0.0019 for all green rows. The corresponding value of b_2, when estimated, was about 10^{-5}.

DBAT	IO parameters				Internal assessment		External assessment
model	b_1	b_2	P	n	σ_0	Point RMS (pixels)	RMSE (μm)
2			P	8	5.9	0.79	82.3
				6	5.9	0.80	92.9
3	b_1	b_2	P	10	1.1	0.15	11.9
	b_1		P	9	1.1	0.15	11.9
	b_1	b_2		8	3.0	0.41	73.2
	b_1			7	3.0	0.41	72.9
		b_2	P	9	5.9	0.79	82.3
		b_2		7	5.9	0.80	92.6
4	b_1	b_2	P	10	1.1	0.15	11.9
	b_1		P	9	1.1	0.15	11.9
	b_1	b_2		8	3.0	0.41	73.3
	b_1			7	3.0	0.41	73.3
		b_2	P	9	5.9	0.79	82.3
		b_2		7	5.9	0.80	92.8

4.2 Camera calibration of a tilt-shift lens

In the second experiment, models 2, 3 and 4 were used to calibrate a camera with a tilt-shift lens. A tilt-shift lens allows the following movements of the lens (Ray, 2002):

- a *tilt*, i.e. a rotation of the optical axis around either the exit pupil or the center of the sensor plane,
- a *shift*, i.e. a translation of the optical axis, and
- a *rotation*, i.e. a rotation about the optical axis.

The data set from Nocerino et al. (2016) was used for the calibration. The data set was acquired by a Nikon D750 full-frame DSLR camera with a PC-E Micro NIKKOR 45mm f/2.8D ED tilt-shift lens (Figure 6) in two different configurations:

NORMAL The normal configuration where neither tilt nor shift was applied.

TILTED The lens was tilted in the vertical plane by 4 degrees.

The calibration target was a 3D photogrammetric calibration test object (Figure 7) with about 170 coded targets with a largest dimension of 900 mm. A high redundancy photogrammetric camera network was realised, consisting of up to 48 convergent images, with a diversity of camera roll angles to enhance the determinability of the IO parameters (Fraser, 2001).

The NORMAL data set was analysed and used as a reference for the processing of the TILTED data set. The TILTED data set was

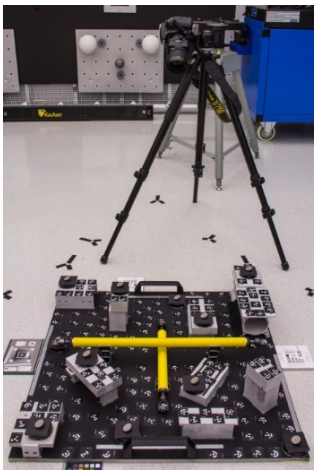

Figure 7: The 3D test object consisted of about 170 coded targets with a largest dimension of 900 mm.

analysed by a self-calibration bundle adjustment in DBAT using models 2, 3 or 4. The datum and prior weights were the same as in the synthetic experiment. Furthermore, the parameters b_1, b_2, and P were individually included or excluded from the estimation (P_1 and P_2 were always estimated together). The quality of the estimation was evaluated as in the synthetic experiment with the results of the NORMAL data set used as the reference. The results are given in Table 2.

From Table 2, we may conclude that the difference between models 3 and 4 on real-world data is small. The internal and external residuals were small when the aspect parameter b_1 and the decentering distortion parameters P_1-P_2 were included in the esti-

mation. In those cases, the estimated b_1 value was about 0.0019. This value is equal to the value in Nocerino et al. (2016, Table 3, column T-4S0R90) that was estimated by other software. In contrast, the inclusion of b_2 had a negligible effect on the residuals.

5. DISCUSSION

In this paper we discuss how the residual computations used by bundle adjustment can be split into modules. If each module is responsible for computing Jacobians with respect to each parameter, in addition to the function value proper, Jacobians of complex expressions can be computed using the chain rule. Furthermore, the analytical Jacobians of each module can be validated independently of the other modules.

The modular technique has previously been used in the Damped Bundle Adjustment Toolbox (DBAT) to model the Photogrammetric and Computer Vision adaptations of the (Brown, 1971) lens distortion models (Börlin and Grussenmeyer, 2016). In this paper, the Photogrammetric pipeline was extended by an affine transformation. Two models with different placement of the affine transformation compared to lens distortion were implemented with minimal effort.

An experiment on synthetic image observations was performed. The conclusion is that the relative placement of the affine transformation and lens distortion does matter and that DBAT was able to distinguish which model was used to generate the synthetic data.

The tilt-shift lens is a complex design whose projection model is not yet rigorously supported by DBAT. In a previous paper, it was found that some of the deviation of the tilt-shift lens from the classic photogrammetric projection model (pin-hole with Brown lens distortion) could be absorbed by including the affine distortion parameter b_1 in the estimation Nocerino et al. (2016). In this paper, the calibration experiment was constructed to test the hypothesis that the *order* of the affine and lens distortion transformations could be modified to explain more of the tilt-shift distortion. The results suggest that the hypothesis was false — compared to other, possibly unmodelled effects, the ordering does not significantly contribute to the observed residuals. The results however support the conclusion in Nocerino et al. (2016) that the aspect parameter b_1 and the decenter parameters P_1-P_2 are significant but the skew parameter b_2 is not.

The primary goal of this paper was to show how the presented modularity of DBAT can be used to test novel projection models. The first experiment shows that the ordering does have an effect and is detectable if it is the dominant factor. The second experiment shows that the tested re-ordering has little effect, thus suggesting that the ordering is not the dominant factor. In both cases, the implementation effort to test the hypothesis was minimal. This suggests that DBAT can be used to test novel projection models in the future. Potential uses of the modularity include replacing a module to form another projection model. For example, the pin-hole module could be swapped by a fish-eye module, or rotation modules could be swapped between omega-phi-kappa and azimuth-tilt-swing.

Future work to better understand the tilt-shift lens include modelling changes in the 3D geometry of the lens with DBAT.

References

Börlin, N. and Grussenmeyer, P., 2013. Bundle adjustment with and without damping. *Photogramm Rec* 28(144), pp. 396–415.

Börlin, N. and Grussenmeyer, P., 2016. External verification of the bundle adjustment in photogrammetric software using the damped bundle adjustment toolbox. *International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences* XLI-B5, pp. 7–14.

Brown, D. C., 1971. Close-range camera calibration. *Photogrammetric Engineering* 37(8), pp. 855–866.

Förstner, W. and Wrobel, B., 2004. *Mathematical Concepts in Photogrammetry*. 5 edn, IAPRS, chapter 2, pp. 15–180.

Fraser, C. S., 2001. Photogrammetric camera component calibration: A review of analytical techniques. In: A. Gruen and T. S. Huang (eds), *Calibration and Orientation of Cameras in Computer Vision*, Springer Series in Information Sciences, Vol. 34, Springer, pp. 95–121.

Heikkilä, J. and Silvén, O., 1997. A four-step camera calibration procedure with implicit image correction. In: *Proc CVPR*, IEEE, San Juan, Puerto Rico, pp. 1106–1112.

Kraus, K., 1993. *Photogrammetry*. Dümmler.

Lucas, J. R., 1963. Differentiation of the orientation matrix by matrix multipliers. *Photogrammetric Engineering* 29(4), pp. 708–715.

Luhmann, T., Robson, S., Kyle, S. and Boehm, J., 2014. *Close-Range Photogrammetry and 3D Imaging*. 2nd edn, De Gruyter.

Magnus, J. R. and Neudecker, H., 2007. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. 3 edn, John Wiley & Sons. Accessed: 2017-08-11.

Mikhail, E. M., Bethel, J. S. and McGlone, J. C., 2001. *Introduction to Modern Photogrammetry*. Wiley.

Nocerino, E., Menna, F., Remondino, F., Beraldin, J.-A., Cournoyer, L. and Reain, G., 2016. Experiments on calibrating tilt-shift lenses for close-range photogrammetry. *International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences* XLI(B5), pp. 99–105.

Ray, S. F., 2002. *Applied Photographic Optics*. 3 edn, Focal Press.

Tsai, R. Y., 1987. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE T Robotic Autom* RA-3(4), pp. 323–344.

Wolf, P. R. and Dewitt, B. A., 2000. *Elements of Photogrammetry*. 3 edn, McGraw-Hill, New York, USA.

Zhang, Z., 2000. A flexible new technique for camera calibration. *IEEE T Pattern Anal* 22(11), pp. 1330–1334.

A. THE CHAIN RULE

According to the chain rule, if a univariate function $h(x)$ is formed as the composition of two univariate, differentiable functions $f(x)$ and $g(x)$ as

$$h(x) = f(g(x)), \tag{11}$$

the derivative $h'(x)$ of the composed function may be calculated as the product of the derivatives $f'(x)$ and $g'(x)$. With substitutions $y = f(u)$, $u = g(x)$, the derivative may be written as

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}, \tag{12}$$

where the re-appearance of the denominator du of one factor as the numerator of the next gives the chain rule its name. To make it clear where each derivative is computed, the computation of $h'(x)$ at $x = c$ is usually written as

$$\left.\frac{dy}{dx}\right|_{x=c} = \left.\frac{dy}{du}\right|_{u=g(c)} \left.\frac{du}{dx}\right|_{x=c}, \tag{13}$$

where the subscript is to be read “evaluated at”.

The chain rule may be extended to multivariate functions. For instance, if

$$g(x(u,v), y(u,v), z(u,v))$$

is a function of (x, y, z) that themselves are functions of (u, v), the chain rule dictates that

$$\frac{\partial g}{\partial u} = \frac{\partial g}{\partial x}\frac{\partial x}{\partial u} + \frac{\partial g}{\partial y}\frac{\partial y}{\partial u} + \frac{\partial g}{\partial z}\frac{\partial z}{\partial u}, \tag{14a}$$

$$\frac{\partial g}{\partial v} = \frac{\partial g}{\partial x}\frac{\partial x}{\partial v} + \frac{\partial g}{\partial y}\frac{\partial y}{\partial v} + \frac{\partial g}{\partial z}\frac{\partial z}{\partial v}. \tag{14b}$$

If the variables are collected in vectors

$$w = \begin{pmatrix} u \\ v \end{pmatrix}, \; t(w) = \begin{pmatrix} x(w) \\ y(w) \\ z(w) \end{pmatrix}, \tag{15}$$

we find that the Jacobian $\left[\frac{dg}{dw}\right]$ at $w = c$

$$\left[\frac{dg}{dw}\right]_{w=c} = \left[\frac{dg}{dt}\right]_{t=t(c)} \left[\frac{dt}{dw}\right]_{w=c} \tag{16}$$

is the product of the two Jacobians $\left[\frac{dg}{dt}\right]$ and $\left[\frac{dt}{dw}\right]$.

B. JACOBIANS

B.1 Preliminaries

This section mostly follows (Magnus and Neudecker, 2007).

B.1.1 The $\mathrm{vec}(\cdot)$ operator Given an $m \times n$ matrix

$$A = \begin{pmatrix} a_1 & \cdots & a_n \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \in \Re^{m\times n}, \tag{17}$$

the vectorisation operator $\mathrm{vec}(\cdot)$ rearranges A into a column vector where the elements of A are in column-major order

$$\mathrm{vec}(A) = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{mn} \end{pmatrix} \in \Re^{mn\times 1}. \tag{18}$$

B.1.2 Jacobians of matrix functions The Jacobian of any scalar- or vector-valued function f with respect to a matrix argument A is implicitly assumed to be with respect to $\mathrm{vec}(A)$, i.e.

$$\left[\frac{df(A)}{dA}\right] \equiv \left[\frac{df(A)}{d\,\mathrm{vec}(A)}\right]. \tag{19}$$

Furthermore, the Jacobian of any matrix-valued function $F(A)$ is implicitly assumed to be applied to $\mathrm{vec}(F(A))$, i.e.

$$\left[\frac{dF(A)}{dA}\right] \equiv \left[\frac{d\,\mathrm{vec}(f(A))}{d\,\mathrm{vec}(A)}\right]. \tag{20}$$

B.1.3 The Kronecker product The *Kronecker product* $C = A \otimes B$, where the matrices $A \in \Re^{m\times n}$, $B \in \Re^{p\times q}$, and $C \in \Re^{mp\times nq}$, is defined as

$$C = A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nn}B \end{pmatrix}. \tag{21}$$

B.1.4 The Jacobian of matrix expressions For matrices $A \in \Re^{k\times l}$, $B \in \Re^{l\times m}$, $C \in \Re^{m\times n}$, the following identities hold:

$$\mathrm{vec}(AB) = (I_m \otimes A)\,\mathrm{vec}(B) \tag{22a}$$

$$= (B^T \otimes I_k)\,\mathrm{vec}(A) \tag{22b}$$

and

$$\mathrm{vec}(ABC) = (C^T \otimes A)\,\mathrm{vec}(B) \tag{23a}$$

$$= (I_n \otimes AB)\,\mathrm{vec}(C) \tag{23b}$$

$$= (C^T B^T \otimes I_k)\,\mathrm{vec}(A). \tag{23c}$$

Equations (22) and (23) can be used to derive Jacobians of matrix products.

B.2 Component functions

B.2.1 The transpose The transpose of an m-by-n matrix A

$$B(A) = A^T \tag{24a}$$

is a permutation of the elements of A. The Jacobian is a permutation matrix known as the *Commutation matrix* K_{mn}

$$\left[\frac{dB}{dA}\right] = K_{mn}. \tag{24b}$$

B.2.2 Translation The translation of a point $p \in \Re^3$ in by an offset $c \in \Re^3$ is given by

$$T_3(p, p_0) = p + p_0. \tag{25a}$$

The Jacobians of T_3 with respect to p and p_0 are

$$\left[\frac{dT_3(p,p_0)}{dp}\right] = \left[\frac{dT_3(p,p_0)}{dp_0}\right] = I_3. \tag{25b}$$

If the translation is applied to m points stored as columns in P, we instead get

$$T_3(P, p_0) = P + p_0 \mathbf{1}_m^T, \tag{25c}$$

and

$$\left[\frac{dT_3(P,p_0)}{dP}\right] = I_m \otimes I_3 = I_{3m} \tag{25d}$$

$$\left[\frac{dT_3(P,p_0)}{dp_0}\right] = \mathbf{1}_m \otimes I_3. \tag{25e}$$

Similarly for a 2D point u and offset u_0,

$$T_2(u, u_0) = u + u_0, \quad \left[\frac{dT_2(u,u_0)}{du}\right] = \left[\frac{dT_2(u,u_0)}{du_0}\right] = I_2. \tag{26}$$

B.2.3 3D linear transformation An arbitrary linear transformation of a point $p \in \Re^3$ can be formulated as

$$L(M,p) = Mp, \tag{27a}$$

with Jacobians given by (22)

$$\left[\frac{dL(M,p)}{dM}\right] = p^T \otimes I_3, \qquad \left[\frac{dL(M,p)}{dp}\right] = M. \tag{27b}$$

B.2.4 3D rotation matrix If the 3D rotation matrix is defined using the $\omega - \phi - \kappa$ Euler angles (Förstner and Wrobel, 2004, eqs. (2.128)–(2.130)), and with the vector $k = \begin{pmatrix}\omega & \phi & \kappa\end{pmatrix}^T$, we have the following rotations

$$R_1(\omega) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{pmatrix}, \tag{28a}$$

$$R_2(\phi) = \begin{pmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{pmatrix}, \tag{28b}$$

$$R_3(\kappa) = \begin{pmatrix} \cos\kappa & -\sin\kappa & 0 \\ \sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{28c}$$

$$R(k) = R(\omega,\phi,\kappa) = R_3(\kappa)R_2(\phi)R_1(\omega), \tag{28d}$$

with Jacobians given by Lucas (1963, eqs. (3)–(9))

$$\left[\frac{dR(k)}{d\omega}\right] = -R(\omega,\phi,\kappa)P_x, \tag{28e}$$

$$\left[\frac{dR(k)}{d\phi}\right] = -R_3(\kappa)R_2(\phi)P_yR_1(\omega), \tag{28f}$$

$$\left[\frac{dR(k)}{d\kappa}\right] = -P_zR(\omega,\phi,\kappa), \tag{28g}$$

$$\left[\frac{dR(k)}{dk}\right] = \begin{pmatrix}\left[\frac{dR(k)}{d\omega}\right] & \left[\frac{dR(k)}{d\phi}\right] & \left[\frac{dR(k)}{d\kappa}\right]\end{pmatrix}. \tag{28h}$$

B.2.5 Pin-hole projection The pin-hole projection of a 3D point p to a 2D point is given by

$$H(p) = \frac{1}{p_3}\begin{pmatrix}p_1 \\ p_2\end{pmatrix}, \tag{29a}$$

$$\left[\frac{dH(p)}{dp}\right] = \frac{1}{p_3}\begin{pmatrix}1 & & -\frac{p_1}{p_3} \\ & 1 & -\frac{p_2}{p_3}\end{pmatrix}. \tag{29b}$$

B.2.6 2D scaling The scaling of the 2D point u by a scalar k is given by

$$S(k,u) = ku. \tag{30a}$$

$$\left[\frac{dS(k,u)}{du}\right] = kI_2, \qquad \left[\frac{dS(k,u)}{dk}\right] = u. \tag{30b}$$

B.2.7 2D affine transformation The affine transformation matrix A is defined as

$$A_m(b) = \begin{pmatrix}1+b_1 & b_2 \\ 0 & 1\end{pmatrix}, \tag{31a}$$

where b are the affine parameters. The corresponding 2D affine transformation function A_2 is defined as

$$A_2(u,b) = A_m(b)u, \tag{31b}$$

$$\left[\frac{dA_2(u,b)}{du}\right] = A_m(b), \qquad \left[\frac{dA_2(u,b)}{db}\right] = \begin{pmatrix}u^T \\ \mathbf{0}_2^T\end{pmatrix}. \tag{31c}$$

B.2.8 Lens distortion

B.2.8.1 Components To simplify the expressions below, we define a number of helper functions: The r function is the norm (“radius”) squared of a vector:

$$r(u) = u^Tu, \qquad \left[\frac{dr(u)}{du}\right] = 2u^T, \tag{32a}$$

The v_p function expands a scalar value x to a vector of its powers:

$$v_p(x;n) = \begin{pmatrix}x \\ x^2 \\ \dots \\ x^n\end{pmatrix}, \quad \left[\frac{dv_p(x;n)}{dx}\right] = \begin{pmatrix}1 \\ 2x^1 \\ \dots \\ nx^{n-1}\end{pmatrix}, \tag{32b}$$

The radial scaling function r_s computes the inner product between a coefficient vector c and the power vector of the radial values squared:

$$r_s(u,c) = c^Tv_p(r(u);|c|), \tag{32c}$$

$$\left[\frac{dr_s(u,c)}{du}\right] = c^T\left[\frac{dv_p(r(u);|c|)}{dr}\right]\left[\frac{dr(u)}{du}\right], \tag{32d}$$

$$\left[\frac{dr_s(u,c)}{dc}\right] = v_p(r(u);|c|)^T, \tag{32e}$$

where $|c|$ is the number of elements of the vector c. Finally, the tangential scaling function t_s computes the non-radial part of the tangential distortion

$$t_s(u,p) = (u^TuI_2 + 2uu^T)p, \tag{32f}$$

$$\left[\frac{dt_s(u,p)}{du}\right] = 2pu^T + 2p^TuI_2 + 2up^T, \tag{32g}$$

$$\left[\frac{dt_s(u,p)}{dp}\right] = u^TuI_2 + 2uu^T. \tag{32h}$$

Given the helper functions above, the radial distortion part of Brown (1971, equation (20)) is reduced to

$$d_r(u,K) = ur_s(u,K), \tag{33a}$$

$$\left[\frac{dd_r(u,K)}{du}\right] = I_2r_s(u,K) + u\left[\frac{dr_s(u,K)}{du}\right], \tag{33b}$$

$$\left[\frac{dd_r(u,K)}{dK}\right] = u\left[\frac{dr_s(u,K)}{dK}\right]. \tag{33c}$$

Furthermore, if the P vector of Brown (1971, equation (20)) is split into the tangential part $P_t = \begin{pmatrix}P_1 & P_2\end{pmatrix}^T$ and radial part $P_r = \begin{pmatrix}P_3 & P_4 & \dots\end{pmatrix}^T$, the tangential distortion part of Brown (1971, equation (20)) becomes

$$d_t(u,P) = t_s(u,P_t)(1 + r_s(u,P_r)), \tag{34a}$$

$$\left[\frac{dd_t(u,P)}{du}\right] = (1 + r_s(u,P_r))\left[\frac{dt_s(u,P_t)}{du}\right] + t_s(u,P_t)\left[\frac{dr_s(u,P_r))}{du}\right], \tag{34b}$$

$$\left[\frac{dd_t(u,P)}{dP_t}\right] = (1 + r_s(u,P_r))\left[\frac{dt_s(u,P_t)}{dP_t}\right], \tag{34c}$$

$$\left[\frac{dd_t(u,P)}{dP_r}\right] = t_s(u,P_t)\left[\frac{dr_s(u,P_r)}{dP_r}\right], \tag{34d}$$

$$\left[\frac{dd_t(u,P)}{dP}\right] = \begin{pmatrix}\left[\frac{dd_t(u,P)}{dP_t}\right] & \left[\frac{dd_t(u,P)}{dP_r}\right]\end{pmatrix}. \tag{34e}$$