Calibration Wizard: A Guidance System for Camera Calibration Based on Modelling Geometric and Corner Uncertainty – Supplementary Material –

Songyou Peng*
ETH Zurich
songyou.peng@inf.ethz.ch

Peter Sturm
INRIA Grenoble – Rhône-Alpes
peter.sturm@inria.fr

In this document, we provide the following details:

- Section 1: provides another synthetic experiment to further demonstrate the effectiveness of our system.
- Section 2: efficient computation of the next best pose (cf. section 2.2 of the main paper) including the incorporation of autocorrelation matrices in this computation (section 3 of the main paper).
- Section 3: full details on the computation of partial derivatives in J (cf. sections 2.1 and 2.2 of the main paper) and a full example of this: pinhole model with radial distortion.
- Section 4: describes a way of visualizing the uncertainty of calibration results through pixel-wise uncertainty maps, displaying expected reprojection errors induced by errors on intrinsic parameters.

1. Another Synthetic Experiment

When users try to calibrate a camera, they tend to randomly move the camera around and get as many images as possible for calibration. With our calibration wizard, we have already shown the superiority over such randomly-captured case in the main paper. Now, considering that calibration is an interactive procedure, i.e., a user can move the camera around, we could actually keep all the intermediate frames when moving from one position to the next, and then all these frames can be used for calibration. Here we perform another synthetic test to validate that the calibration can still be improved by moving to optimal poses under this scenario. Provided the frame rate is 25 fps and it takes 1 second to move from one pose to the next, 25 images can then be acquired within *one path*. To this end, the experimental process is as follows.

First, we create 4 random poses using exactly the same way described in the main paper, from which we could acquire 3 paths by pose interpolation. Thus, there are $3 \times 25 = 75$ frames in total for the case "Random". This process mimics moving the camera from one pose to another continuously for three times.

Then, we randomly take 5 images within the first path to get the initial calibration for our wizard. The wizard then proposes a next best pose and while the simulated camera moves there, 25 additional images for calibration are acquired. This is done 3 times, for 75 additional calibration images in total. This input leads to the results labeled as "Wizard" in the following. The same process is also applied for taking autocorrelation matrix into account in next best pose computation ("Wizard-Auto")

Finally, we simply compare the calibration results acquired from random-generated paths and wizard-generated ones. As illustrated in Fig. 1, we can clearly notice that the calibration results from our system have higher precision and accuracy than the results from random paths.

We also perform the comparison with respect to the level of noise added to 2D corner points. In this experiment, zero-mean Gaussian noise with standard deviation of $0.1, 0.2, 0.5, 1$ respectively 2 pixels has been added to corner points, and the comparisons are shown in Fig. 2. Again, even when unrealistically strong noise is added ($\sigma = 2$), our system provides much better estimation of the focal length. Moreover, when considering the autocorrelation matrices, our system seems more robust to the noise.

Note that we only select the results for some intrinsic parameters for these two experiments, but similar results can be expected for other parameters like principal points.

2. An efficient way of computing the covariance matrix of intrinsic parameters

As already explained in the paper, the Jacobian matrix J can be denoted as:

$$J = \begin{pmatrix} A_1 & B_1 & 0 & \cdots & 0 \\ A_2 & 0 & B_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_m & 0 & 0 & \cdots & B_m \end{pmatrix} \tag{1}$$

*Most work was done while he was an intern at INRIA.

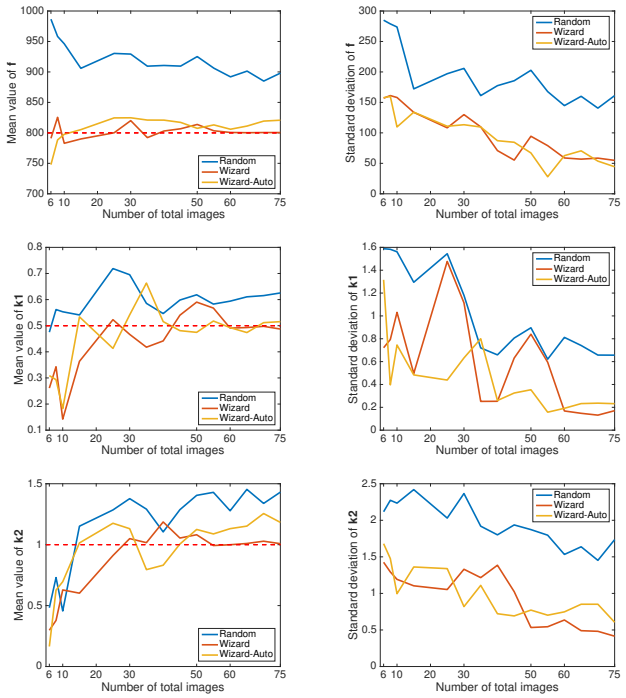


Figure 1. Comparisons of the intrinsic parameters estimated from three schemes on synthetic data: the paths generated from random poses, generated from the poses proposed by calibration wizard without and with autocorrelation matrix. $f = 800, (u, v) = (320, 240), k_1 = \mathbf{0.5}, k_2 = \mathbf{1}$. Red dashed lines represent the ground truth values. Wizard images achieve superior performance over random images on all intrinsic parameters.

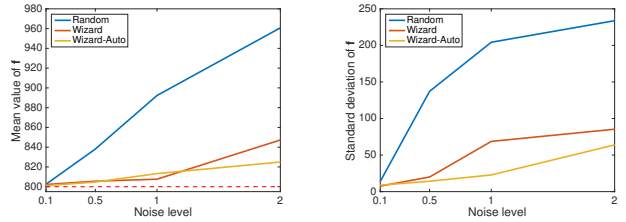


Figure 2. Comparisons of the calibration schemes concerning the robustness to added noise. Zero-mean Gaussian noise with standard deviation of $0.1, 0.2, 0.5, 1$ respectively 2 pixels is added to the 2D target points. The intrinsic parameters estimated from the wizard-generated paths (shown are results for the focal length), are more accurate (left) and precise (right) than random-generated paths, especially when the noise level is high. The total number of image is 75.

Let us examine in detail the incorporation of the autocorrelation matrices of image corner points in the computation of the information matrix, as shown in Eq. (9) in the paper:

$$H = J^\top \text{diag}(C_{11}, C_{12}, \cdots, C_{1n}, C_{21}, \cdots, C_{mn}) J \quad (2)$$

Let us decompose the matrices A_i and B_i appearing in the definition of the J, into matrices A_{ij} and B_{ij} containing the residual associated with a single image point each (point j in image i):

$$A_{ij} = \begin{pmatrix} \frac{\partial \hat{x}_{ij}}{\Theta_1} & \cdots & \frac{\partial \hat{x}_{ij}}{\Theta_k} \\ \frac{\partial \hat{y}_{ij}}{\Theta_1} & \cdots & \frac{\partial \hat{y}_{ij}}{\Theta_k} \end{pmatrix} \quad B_{ij} = \begin{pmatrix} \frac{\partial \hat{x}_{ij}}{\Pi_{i,1}} & \cdots & \frac{\partial \hat{x}_{ij}}{\Pi_{i,6}} \\ \frac{\partial \hat{y}_{ij}}{\Pi_{i,1}} & \cdots & \frac{\partial \hat{y}_{ij}}{\Pi_{i,6}} \end{pmatrix}$$

The A_{ij} are of size $2 \times k$ (k is the number of intrinsic parameters) and the B_{ij} of size 2×6. We then have:

$$A_i = \begin{pmatrix} A_{i1} \\ \vdots \\ A_{in} \end{pmatrix} \quad B_i = \begin{pmatrix} B_{i1} \\ \vdots \\ B_{in} \end{pmatrix}$$

Now, the information matrix H of Eq. (2) can be structured as:

$$H = \begin{pmatrix} U & W \\ W^\top & V \end{pmatrix} \quad (3)$$

where

$$U = \sum_i U_i \quad (4)$$

$$\text{with } U_i = \sum_j A_{ij}^\top C_{ij} A_{ij} \quad (5)$$

$$W = \begin{pmatrix} W_1 & \cdots & W_m \end{pmatrix} \quad (6)$$

$$\text{with } W_i = \sum_j A_{ij}^\top C_{ij} B_{ij} \quad (7)$$

$$V = \begin{pmatrix} V_1 & & & \\ & V_2 & & \\ & & \ddots & \\ & & & V_m \end{pmatrix} \quad (8)$$

$$\text{with } V_i = \sum_j B_{ij}^\top C_{ij} B_{ij} \quad (9)$$

U is a symmetric matrix of size $k \times k$, W is of size $k \times (6m)$ (remember that m is the number of images) and V is a block-diagonal matrix consisting of m symmetric 6×6 matrices V_i.

As described in [1], the upper-left sub-matrix of H^{-1} is given by

$$\Sigma = (U - WV^{-1}W^\top)^+ \quad (10)$$

which is a pseudo-inverse of a $k \times k$ symmetric matrix. Let us expand this using the above definitions of U, V and W.

Using the above definitions of U, V and W, we can rewrite Σ as:

$$\Sigma = \left\{ \sum_i \left(U_i - W_i V_i^{-1} W_i^\top \right) \right\}^+ \quad (11)$$

The computation is efficient. The V_i are symmetric and of size 6×6, hence their inversion is a small problem. Other than that, the computation only involves products and sums

of small matrices and one final inversion of a $k \times k$ symmetric matrix.

Let us now consider the computation of the next best pose. As explained in the paper, we use global optimization algorithms for this purpose. They will need to compute Σ and its trace multiple times, for multiple hypothetical next poses. It is thus interesting to study how to cut computation times by performing adequate pre-computations. This is very simple in our case. Let image $m+1$ be the next image, for which to compute the optimal pose. Images 1 till m are thus already acquired and we have computed intrinsic and pose parameters for them. All we need to do is to compute once the following part of equation (11), for images 1 to m:

$$A = \sum_{i=1}^{m} \left(U_i - W_i V_i^{-1} W_i^\top \right) \tag{12}$$

Then, for each hypothetical pose for image $m+1$, we only need to compute

$$\Sigma = \left\{ A + U_{m+1} - W_{m+1} V_{m+1}^{-1} W_{m+1}^\top \right\}^+ \tag{13}$$

This computation is efficient: besides the computation of partial derivatives of the projection function for the hypothetical new pose (see section 3 for details) and the sum or multiplication of small matrices (of size at most $(max(k,6) \times max(k,6))$, it only involves the inversion of a symmetric matrix of size 6×6 and of a symmetric matrix of size $k \times k$.

Note that in this section we handled the incorporation of autocorrelation matrices. If one wishes to work without them, it suffices to replace the C_{ij} by identity matrices.

3. Details on the computation of partial derivatives in J

In the following, we use the following shorthand notations for partial derivatives, which makes for easier reading. If A is a vector of size a and B a vector of size b (with possibly, $a = 1$ or $b = 1$), then we write:

$$\frac{\partial A}{\partial B} = \begin{pmatrix} \frac{\partial A_1}{\partial B_1} & \cdots & \frac{\partial A_1}{\partial B_b} \\ \vdots & \ddots & \vdots \\ \frac{\partial A_a}{\partial B_1} & \cdots & \frac{\partial A_a}{\partial B_b} \end{pmatrix} \tag{14}$$

Also, if A is a matrix and B a scalar, then $\frac{\partial A}{\partial B}$ is the matrix gathering the partial derivatives of the coefficients of A relative to the scalar B, in the same order as they appear in A.

As we have seen in the paper, $3D$ to $2D$ projections can be written as:

$$p_x(\Theta, \Pi, Q) = q_x(\Theta, RQ + t) = q_x(\Theta, S) \tag{15}$$
$$p_y(\Theta, \Pi, Q) = q_y(\Theta, RQ + t) = q_y(\Theta, S) \tag{16}$$

Thanks to the chain rule, the derivatives of residuals $\hat{x}$ (for their definition, see section 2 of the main paper) with respect to extrinsic parameters can be decomposed as:

$$\frac{\partial \hat{x}}{\partial R} = \frac{\partial p_x}{\partial R} = \frac{\partial q_x}{\partial S} \frac{\partial S}{\partial R} \tag{17}$$
$$\frac{\partial \hat{x}}{\partial t} = \frac{\partial p_x}{\partial t} = \frac{\partial q_x}{\partial S} \frac{\partial S}{\partial t} \tag{18}$$

and likewise for partial derivatives of $\hat{y}$. Here, we use the informal notation

$$\partial R = \partial \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \tag{19}$$

where the three angles α, β, γ parameterize the rotation matrix R as shown on page 2 of the main paper and as reproduced here:

$$R = R_z(\gamma) R_y(\beta) R_x(\alpha) \tag{20}$$

$\partial q_x / \partial S$ and $\partial q_y / \partial S$ depend on the projection functions for the camera model used for calibration (see below for an example, the pinhole model with one radial distortion coefficient), while $\partial S / \partial R$ and $\partial S / \partial t$ are generic for all camera models. Remember that

$$S = RQ + t \tag{21}$$

Thus:

$$\frac{\partial S}{\partial R} = \begin{pmatrix} \frac{\partial S_1}{\partial \alpha} & \frac{\partial S_1}{\partial \beta} & \frac{\partial S_1}{\partial \gamma} \\ \frac{\partial S_2}{\partial \alpha} & \frac{\partial S_2}{\partial \beta} & \frac{\partial S_2}{\partial \gamma} \\ \frac{\partial S_3}{\partial \alpha} & \frac{\partial S_3}{\partial \beta} & \frac{\partial S_3}{\partial \gamma} \end{pmatrix} = R \begin{pmatrix} Q_1 & Q_3 & -Q_2 \\ -Q_3 & Q_2 & Q_1 \\ Q_2 & -Q_1 & Q_3 \end{pmatrix} \tag{22}$$

$$\frac{\partial S}{\partial t} = \begin{pmatrix} \frac{\partial S_1}{\partial t_1} & \frac{\partial S_1}{\partial t_2} & \frac{\partial S_1}{\partial t_3} \\ \frac{\partial S_2}{\partial t_1} & \frac{\partial S_2}{\partial t_2} & \frac{\partial S_2}{\partial t_3} \\ \frac{\partial S_3}{\partial t_1} & \frac{\partial S_3}{\partial t_2} & \frac{\partial S_3}{\partial t_3} \end{pmatrix} = I_{3\times 3} \tag{23}$$

The derivation is straightforward.

3.1. Example: pinhole model with one radial distortion coefficient

Here we use the pinhole model with one radial distortion coefficient as an example. It can be easily generalized to other more complicated models. The model can be represented as:

$$q_x(S) = u + (1 + k_1 r^2) f \frac{S_1}{S_3} \tag{24}$$
$$q_y(S) = v + (1 + k_1 r^2) f \frac{S_2}{S_3} \tag{25}$$

where $r = \sqrt{(\frac{S_1}{S_3})^2 + (\frac{S_2}{S_3})^2} = \frac{1}{S_3}\sqrt{S_1^2 + S_2^2}$. We define $\Theta = (f, u, v, k_1)$.

First, the partial derivatives of the local projection functions with respect to S can be written as:

$$\begin{pmatrix} \frac{\partial q_x}{\partial S} \\ \frac{\partial q_y}{\partial S} \end{pmatrix} = \begin{pmatrix} \frac{\partial q_x}{\partial S_1} & \frac{\partial q_x}{\partial S_2} & \frac{\partial q_x}{\partial S_3} \\ \frac{\partial q_y}{\partial S_1} & \frac{\partial q_y}{\partial S_2} & \frac{\partial q_y}{\partial S_3} \end{pmatrix} \tag{26}$$

where

$$\begin{aligned} \frac{\partial q_x}{\partial S_1} &= \frac{f}{S_3} + \frac{fk_1}{S_3^3}(3S_1^2 + S_2^2) \\ \frac{\partial q_x}{\partial S_2} &= \frac{2fk_1 S_1 S_2}{S_3^3} \\ \frac{\partial q_x}{\partial S_2} &= -\frac{fS_1}{S_3^2} - \frac{3fk_1 S_1(S_1^2 + S_2^2)}{S_3^4} \\ \frac{\partial q_y}{\partial S_1} &= \frac{2fk_1 S_1 S_2}{S_3^3} \\ \frac{\partial q_y}{\partial S_2} &= \frac{f}{S_3} + \frac{fk_1}{S_3^3}(S_1^2 + 3S_2^2) \\ \frac{\partial q_y}{\partial S_2} &= -\frac{fS_2}{S_3^2} - \frac{3fk_1 S_2(S_1^2 + S_2^2)}{S_3^4} \end{aligned}$$

This, together with Eq. (22) and (23), allows to compute the partial derivatives of residuals relative to extrinsic parameters in the Jacobian matrix J, by inserting into Eq. (17) and (18) (and likewise for partial derivatives of $\hat{y}$).

As for the partial derivatives relative to intrinsic parameters, they can be computed as:

$$\frac{\partial \hat{x}}{\partial \Theta} = \frac{\partial q_x}{\partial \Theta} = \begin{pmatrix} (1 + k_1 r^2)\frac{S_1}{S_3} & 1 & 0 & r^2 f \frac{S_1}{S_3} \end{pmatrix} \tag{27}$$

$$\frac{\partial \hat{y}}{\partial \Theta} = \frac{\partial q_y}{\partial \Theta} = \begin{pmatrix} (1 + k_1 r^2)\frac{S_2}{S_3} & 0 & 1 & r^2 f \frac{S_2}{S_3} \end{pmatrix} \tag{28}$$

Now, we have everything needed for building J.

4. Uncertainty map

Here we introduce the concept of uncertainty map which could be an effective tool to visualize the quality of the current calibration. Also, the uncertainty map can be used to evaluate the evolution of the quality along the calibration process when adding more and more images.

Given the $k \times k$ covariance matrix Σ of intrinsic parameters, an uncertainty map is defined as the expected uncertainties of the local projection model across the image plane. That is to say, we propagate the uncertainties of intrinsic parameters to each pixel on the image area. The quality of calibration at any stage of the calibration process can be visualized as shown in Fig. 3.

Mathematically speaking, for each pixel (x, y) on the image, the uncertainty propagation process can be formulated as:

$$\Gamma(x,y) = \begin{pmatrix} \frac{\partial q_x(\Theta,S)}{\partial \Theta}|_{S=S(x,y,\Theta)} \\ \frac{\partial q_y(\Theta,S)}{\partial \Theta}|_{S=S(x,y,\Theta)} \end{pmatrix} \Sigma \begin{pmatrix} \frac{\partial q_x(\Theta,S)}{\partial \Theta}|_{S=S(x,y,\Theta)} \\ \frac{\partial q_y(\Theta,S)}{\partial \Theta}|_{S=S(x,y,\Theta)} \end{pmatrix}^\top \tag{29}$$

where Γ is a 2×2 covariance matrix expressing the uncertainty per image point (x, y). To compute this, we first need to back-project the image point to a 3D point S (any 3D point along the line of sight associated with the image point will do). We denote this as $S(x, y, \Theta)$ in the above equation: the back-projection depends on the image point coordinates x and y and of course on the intrinsic parameters Θ. Then, we propagate the uncertainty on the intrinsic parameters (covariance matrix Σ) in a standard way through the forward projection, as shown in the above equation, to obtain the covariance matrix Γ.

Note that, we are aware that minimizing over this pixel reprojection uncertainty is better than over the trace of the covariance matrix Σ. The main problem with minimizing reprojection uncertainty is computation time, incompatible with real-time. In this paper we stuck to the trace, a commonly used simple cost function. In the future, we will consider the reprojection uncertainty for a subset of pixels.

In the following, we provide an example for the process of acquiring the uncertainty map and some analysis.

4.1. Example: 3-parameter pinhole model

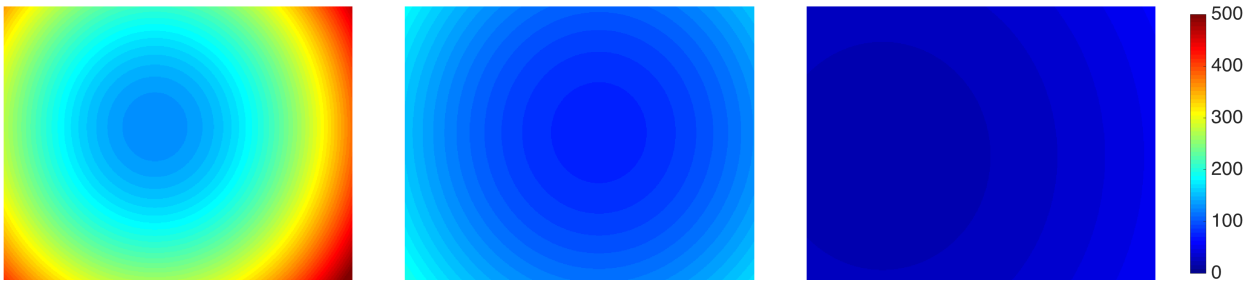
For this simple camera model, the uncertainty map can be analyzed theoretically, as shown in the following. We remind the definition of the 3-parameter pinhole model, where the vector of intrinsic parameters is $\Theta = (f, u, v)^\top$.

$$\begin{aligned} q_x(\Theta, S) &= u + f\frac{S_1}{S_3} \\ q_y(\Theta, S) &= v + f\frac{S_2}{S_3} \end{aligned}$$

These are the "forward" projection equations. As explained above, to construct the uncertainty map, we should back-project each pixel to 3D, i.e. to some 3D point S which, if forward-projected, gives rise to the original pixel. One possibility for the back-projection is as follows:

$$S(x, y, \Theta) = \begin{pmatrix} x - u \\ y - v \\ f \end{pmatrix}$$

We now compute the partial derivatives appearing in



(a) (b) (c)

Figure 3. Illustrations for the uncertainty maps and the effectiveness of our camera wizard. The size of the map is the same as acquired image size 640×480. (a) Initial calibration results with 3 freely taken images. On top of the 3 freely-taken images, (b) add two freely taken images. (c) add two images proposed by wizard. We can clearly visualize: (1) the more calibration images, the lower the uncertainty (2) the uncertainty obtained from using wizard images is much lower than with freely taken ones (cf. the scale on the right hand side).

Eq. (29):

$$\begin{pmatrix} \frac{\partial q_x(\Theta,S)}{\partial \Theta}|_{S=S(x,y,\Theta)} \\ \frac{\partial q_y(\Theta,S)}{\partial \Theta}|_{S=S(x,y,\Theta)} \end{pmatrix} = \begin{pmatrix} \frac{S_1}{S_3} & 1 & 0 \\ \frac{S_2}{S_3} & 0 & 1 \end{pmatrix} |_{S=S(x,y,\Theta)}$$
$$= \begin{pmatrix} \frac{x-u}{f} & 1 & 0 \\ \frac{y-v}{f} & 0 & 1 \end{pmatrix}$$

Inserting this in Eq. (29), we get the desired covariance matrices per image point:

$$\Gamma(x,y) = \begin{pmatrix} \frac{x-u}{f} & 1 & 0 \\ \frac{y-v}{f} & 0 & 1 \end{pmatrix} \Sigma \begin{pmatrix} \frac{x-u}{f} & \frac{y-v}{f} \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

To analyze the nature of the uncertainty map, we can imagine it as the coefficients of an "uncertainty ellipse" centered at every image pixel. Once we acquire Γ for each pixel, the trace, larger eigenvalue or determinant of Γ can be used to measure the impact of the uncertainty of intrinsic parameters, on this pixel. In Fig. 3, we generate the map by creating an image where the value of every pixel is given by the trace of Γ.

We may further analyze the nature of these uncertainty maps as follows. When computing the trace of Γ in detail, we get:

$$tr(\Gamma(x,y)) = \Sigma_{22}+\Sigma_{33} + \frac{\Sigma_{11}}{f^2}\left[(x-u)^2+(y-v)^2\right] + \frac{2}{f}\left[\Sigma_{12}(x-u)+\Sigma_{13}(y-v)\right]$$

This is quadratic with respect to x and y. The uncertainty map, if visualized in 3D (as height map above the $x-y$-plane), can be shown to be a circular paraboloid. The global minimum of the trace is attained at

$$x^* = u - f\frac{\Sigma_{12}}{\Sigma_{11}} \qquad y^* = v - f\frac{\Sigma_{13}}{\Sigma_{11}}$$

and has the following value:

$$\Sigma_{22} + \Sigma_{33} - \frac{\Sigma_{12}^2 + \Sigma_{13}^2}{\Sigma_{11}}$$

A few observations are as follows. If the covariance between the focal length and the principal point coordinates (Σ_{12} and Σ_{13}) approaches zero, then the minimum of the uncertainty map coincides with the principal point and its value there is the sum of the variances of the principal point coordinates. And if the variance of the focal length tends to zero, the uncertainty map tends to being uniform. Therefore, when the calibration result gets more accurate (low uncertainty), the range of the uncertainty map will become smaller, and its minimum will be approaching the principal point. Finally, an empirical observation is that the covariance matrices Γ, when visualized as uncertainty ellipses, tend to be oriented towards the minimum of the uncertainty map (i.e. at each image point, the associated uncertainty ellipse has an axis that "points" towards the minimum of the uncertainty map).

More analysis, also for more complex camera models, is possible.

In summary, we believe that this concept is a principled way of displaying and interpreting the uncertainty of a calibration estimate.

References

[1] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003. 2