Scientific Experimentation and Evaluation

Arka Mallick, Daiem Ali,Chun Kwang Tan

Hochschule Bonn-Rhien-Sieg

03.05.2015

Setup description:

- To test the camera calibration we are using Microsoft Lifecam 2 camera.
- We have mounted the camera on a tripod to ensure its stability.
- The checkboard pattern used for the calibration is mounted on a hard board to ensure it is almost plane.
- For recording the images we presented the pattern to the camera in random orientations and distances. These orientations are in:
  - X-axis of the image
  - Y-axis of the image
  - Z-axis of of the image
- We used OpenCV Camera Calibration and 3D Reconstruction Toolbox to filter out suitable images for the extraction of the calibration parameters for the camera.
- On the captured images we
  - Extract corners (7,6) pattern rig size.
  - We locate those corners in subpixel precision using openCV feature detection toolbox
  - Using the located corner points the camera calibration function returns us the intrinsic camera parameters as follows:
    - Projection error
    - Camera intrinsic parameters
    - Distortion Matrix
    - Rotation and Translation matrices

* The details of the parameters can be found in [1].

- The algorithms used in the openCV are based on Z.Zhang's work[2] and Bouguet's MATLAB calibration tool.[3]
- Pitfalls:
  - Random orientations and distances make it hard to reproduce results
  - Lighting conditions
  - Motion blur due to shaky human hands presenting pattern

Short Theory of camera calibration

For camera calibration purposes, we are interested in extracting metric information from 2D images. This is usually an important component in 3D computer vision, as in general, camera views are in 2D. This section presents the theory behind camera calibration from Zhang (2000) [2].

From Zhang (2000) [2], they define the relationship between a 3D point, **M**, and its image projection, **m,** as:1

$$s\tilde{m} = A\,[\boldsymbol{R}\ t]\tilde{\mathrm{M}} \qquad (1)$$

where s is an arbitrary scale factor, (R, t) are extrinsic parameters comprising of the rotation (R) and translation (t) which relates the world coordinate system to the camera coordinate system. A is the camera intrinsic matrix, given as:

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

with the (u_0, v_0) being the coordinates of the principal point, α and β the sclae factors in image u and v axes, and the γ parameter describing the skewness of the image axes.

Homography between model plane and image

They assume that the model plane is on Z = 0 of the world coordinate system. This means that they assume that a camera is always looking along the Z-axis of the world coordinate system. This gives the equation:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A\,[r_1 \quad r_2 \quad r_3 \quad t]\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$$

$$= A[r_1 \quad r_2 \quad t]\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Thus, by relating it to Eqn (1), we get:

$$s\tilde{m} = H\tilde{\mathrm{M}} \text{ where}$$

$$H = A[r_1 \quad r_2 \quad t] \qquad (2)$$

Constraints on intrinsic parameters

By denoting $H = [h_1 \quad h_2 \quad h_3]$, and equating it to Eqn (2), we get:

$$[h_1 \quad h_2 \quad h_3] = \lambda A[r_1 \quad r_2 \quad t]$$

where λ is an arbitrary scalar. By exploiting the knowledge that $r_1\ and\ r_2$ are orthonormal, we get:

$$h_1^T A^{-T} A^{-1} h_2 = 0 \qquad (3)$$

$$h_1^T A^{-T} A^{-1} h_1 = h_2^T A^{-T} A^{-1} h_2 \qquad (4)$$

Zhang (2000) [2] noted that $A^{-T}A^{-1}$ actually describes the image of absolute conic (IAC), which is a concept that describes how the relative position to a moving camera is camera, similar to the impression that the moon is somehow following you when driving on a straight road Pollefeys (2002) [5]. Details of the proof can be found in Zhang (2000) [2].

With the above concepts and notations, Zhang (2000) [2] is able to derive a closed-form solution, which estimates the extrinsic parameters.

Also, Zhang (2000) [2] includes a method to estimate radial distortion of the camera lens and incorporates it into his method of camera calibration. For more mathematical proofs, please refer to Zhang (2000) [2] for details.

Estimation of number of images:

- According to the paper by Zhang (2000) [2] we can see any number of images more than 2 are sufficient for the purpose.
- From his paper we can see the absolute and relative error in the calibration process decreases with the number images. So we are using 19 images here.

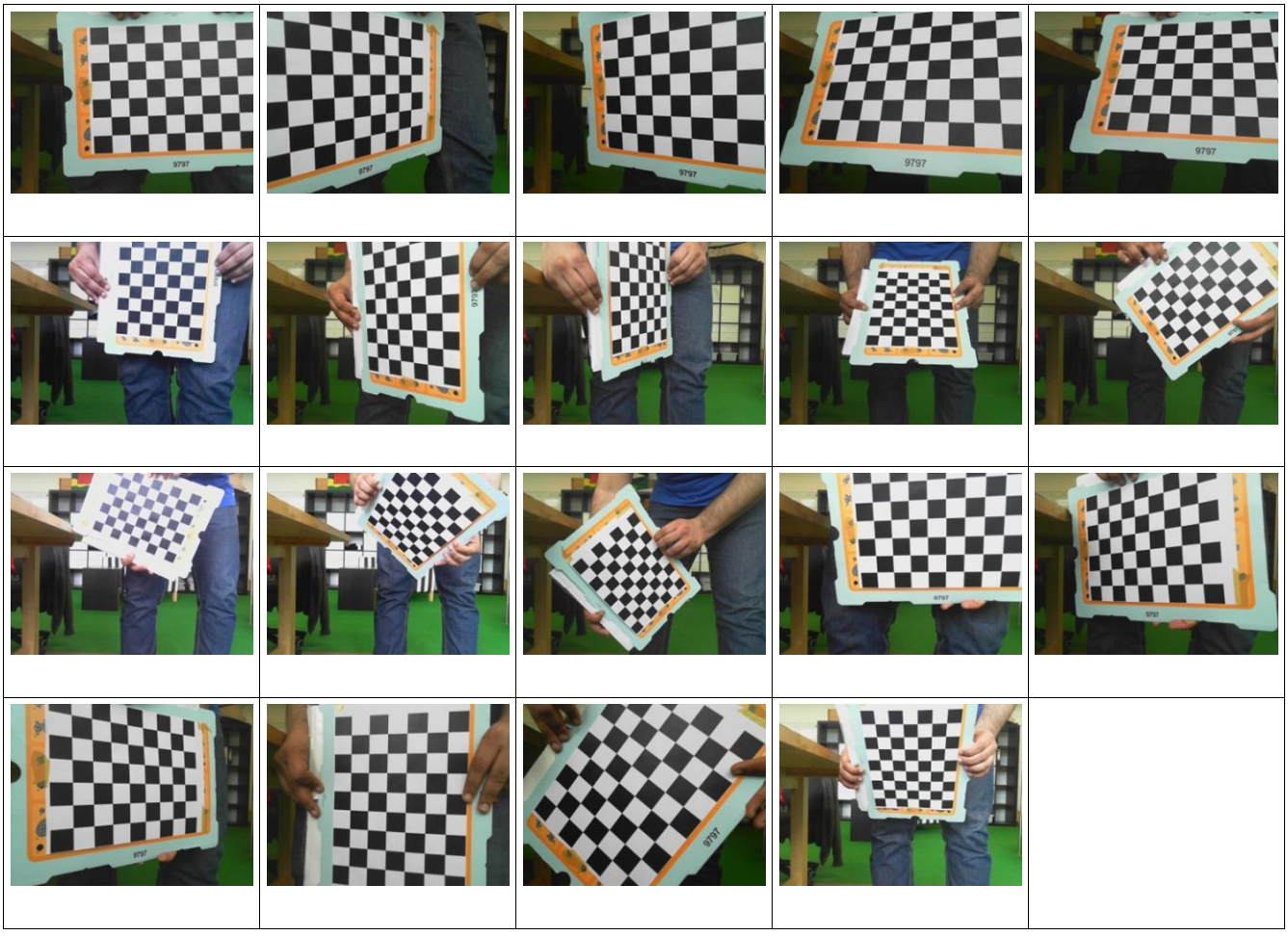


Table 1: The images used for the calibration experiment

Results:

```
camera_matrix : [[ 641.18595954    0.          319.93339088]
                 [   0.          643.03128128  237.02721474]
                 [   0.            0.            1.        ]]
distortion_matrix : [[-0.00876629  0.00358646  0.0024546   0.00232463  0.18954188]]
rotation_vector : [array([[ 0.87676463],
       [ 0.04105643],
       [ 2.95587207]]), array([[-0.45075859],
       [ 0.64548424],
       [-1.50274912]]), array([[-1.12552049],
       [-0.12863679],
       [ 2.88248399]]), array([[-0.00560095],
       [ 0.21992057],
       [-1.4998973 ]]), array([[-0.39972499],
       [ 0.27403736],
       [ 0.05432795]]), array([[ 0.19393173],
       [ 0.1626148 ],
       [ 1.58653682]]), array([[-0.57864604],
       [-0.51862647],
       [-1.45771198]]), array([[ 0.08268645],
       [ 0.55069356],
       [-0.01525055]]), array([[-0.0298304 ],
       [-0.00823022],
       [-0.0364957 ]]), array([[-0.03866544],
       [-0.50550458],
       [ 0.84445444]]), array([[-0.03607791],
       [-0.60003236],
       [-0.02001124]]), array([[ 0.44803243],
       [ 0.89317329],
       [-0.76572702]]), array([[ 0.64646596],
       [-0.45848564],
       [ 1.47739809]]), array([[ 0.83062463],
       [-0.60525191],
       [-1.53730716]]), array([[-0.49260502],
       [ 0.18892639],
       [ 0.07012203]])]
translation_vectors : [array([[  2.25601642],
       [  1.0486905 ],
```

```
       [ 11.90403175]]), array([[ -0.72793866],
       [  1.75198822],
       [ 16.94895204]]), array([[  1.33897648],
       [  1.47580036],
       [ 14.13254336]]), array([[ -0.35167599],
       [ -0.77285935],
       [ 20.16813225]]), array([[ -2.32049184],
       [ -3.7511956 ],
       [ 13.25446363]]), array([[  2.9140143 ],
       [ -2.99315711],
       [ 11.83220418]]), array([[ -1.67613633],
       [  0.25543029],
       [ 18.97887334]]), array([[ -1.43159162],
       [ -3.31162209],
       [ 12.91141542]]), array([[ -1.27936387],
       [ -3.13902925],
       [ 13.40792101]]), array([[ -0.99229247],
       [ -3.01527833],
       [ 21.04239562]]), array([[ -2.699155  ],
       [ -3.49777135],
       [ 10.76897551]]), array([[ -1.01581429],
       [ -4.04265497],
       [ 23.79207841]]), array([[  3.14826567],
       [ -4.7940019 ],
       [ 16.58552334]]), array([[ -1.6388087 ],
       [  0.45292595],
       [ 17.61594671]]), array([[ -0.77285828],
       [ -3.84894118],
       [ 14.43294569]])]
```

Comments:

The translation and rotation vectors calculated here are for each of the 19 images.

Compare the findings with camera calibration toolbox from matlab:

This same exercise was again evaluated using camera toolbox from matlab. The main idea was to evaluate the results with the one obtained from the openCV toolbox. There are some differences in the Matlab implementation by Bouguet (2013) [3], as compared to Zhang (2000) [2]. They used the intrinsic model of Heikkilä and Silven (1997) [4] and includes two extra distortion coefficients corresponding to tangential distortion. Also, Bouguet (2013) [3] explicitly used orthogonality of vanishing points to estimate internal parameters from the homographies and do not provide an initial estimate of distortion coefficients at the initialization phase.

The findings are as follows:

Calibration results after optimization (with uncertainties):

Focal Length: fc = [637.57086 639.96654] ± [4.26265 4.03072]

Principal point: cc = [318.84827 235.79983] ± [5.13144 5.05098]

Skew: alpha_c = [0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees

Distortion: kc = [-0.02302 0.07777 0.00082 0.00253 0.00000] ± [0.01828 0.06461 0.00273 0.00278 0.00000]

Pixel error: err = [0.39578 0.26959] (Extrinsic Error)

The individual results for each of the 19 images are as follows:

Image 1:

Translation vector: Tc_ext = [-37.668065 -93.392110 400.571988]

Rotation vector: omc_ext = [2.240809 2.161362 -0.031408]

Rotation matrix: Rc_ext = [0.036168 0.999333 0.005002

0.998766 -0.035976 -0.034240

-0.034038 0.006234 -0.999401]

Pixel error: err = [0.21496 0.21831]

Image 2:

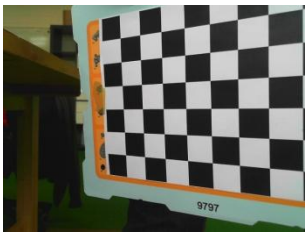
Translation vector: Tc_ext = [-115.664370 -89.503585 267.211396]

Rotation vector: omc_ext = [1.857275 1.771619 0.782928]

Rotation matrix: Rc_ext = [0.011761 0.737713 0.675012
0.995775 -0.070120 0.059284
0.091066 0.671463 -0.735421]

Pixel error: err = [0.45605 0.13332]

Image 3:



Translation vector: Tc_ext = [-42.359536 -98.589887 385.762584]

Rotation vector: omc_ext = [-1.894846 -1.864543 0.452080]

Rotation matrix: Rc_ext = [0.036871 0.852262 -0.521815
0.996611 0.007063 0.081957
0.073535 -0.523068 -0.849113]

Pixel error: err = [0.37702 0.17037]

Image 4:

Translation vector: Tc_ext = [-68.987123 -111.750398 396.014539]

Rotation vector: omc_ext = [2.037183 2.154067 -0.723638]

Rotation matrix: Rc_ext = [-0.106591 0.961665 -0.252663
0.919162 -0.001605 -0.393876

-0.379182 -0.274222 -0.883755]

Pixel error: err = [0.24144 0.32154]

Image 5:

Translation vector: Tc_ext = [-22.411022 -114.618266 431.197649]

Rotation vector: omc_ext = [1.935283 2.081310 -0.704977]

Rotation matrix: Rc_ext = [-0.113592 0.979885 -0.164081

0.877896 0.021668 -0.478361

-0.465183 -0.198384 -0.862698]

Pixel error: err = [0.23168 0.32442]

Image 6:

Translation vector: Tc_ext = [-9.588368 -21.865580 601.419235]

Rotation vector: omc_ext = [-3.000885 -0.097795 0.213870]

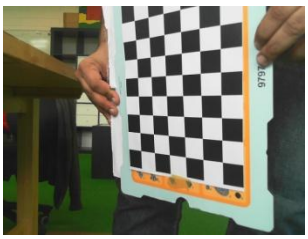
Rotation matrix: Rc_ext = [0.987845 0.055183 -0.145316

0.073817 -0.989263 0.126131

-0.136796 -0.135325 -0.981312]

Pixel error: err = [0.31432 0.32432]

Image 7:



Translation vector: Tc_ext = [-20.997596 53.530755 506.390419]

Rotation vector: omc_ext = [-2.836645 0.007685 1.037703]

Rotation matrix: Rc_ext = [0.764794 -0.046261 -0.642612

0.036738 -0.992665 0.115183

-0.643227 -0.111699 -0.757485]

Pixel error: err = [0.74821 0.21291]

Image 8:

Translation vector: Tc_ext = [-48.324040 14.617894 526.043324]

Rotation vector: omc_ext = [-2.698262 0.130263 -1.316968]

Rotation matrix: Rc_ext = [0.613982 -0.017941 0.789116

-0.136999 -0.986990 0.084153

0.777340 -0.159776 -0.608452]

Pixel error: err = [0.73977 0.09128]

Image 9:

Translation vector: Tc_ext = [-40.921188 13.364572 647.895299]

Rotation vector: omc_ext = [-2.427617 0.025679 0.158880]

Rotation matrix: Rc_ext = [0.992302 -0.061027 -0.107764

0.023973 -0.759047 0.650594

-0.121502 -0.648169 -0.751741]

Pixel error: err = [0.15339 0.46581]

Image 10:

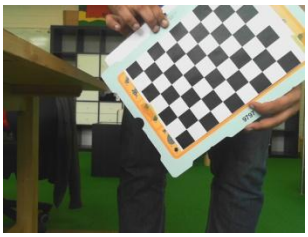
Translation vector: Tc_ext = [-49.349027 8.748681 565.953704]

Rotation vector: omc_ext = [2.432808 0.054410 -0.051243]

Rotation matrix: Rc_ext = [0.998340 0.053008 -0.022503
0.025637 -0.759021 -0.650562
-0.051566 0.648905 -0.759120]

Pixel error: err = [0.10409 0.50543]

Image 11:



Translation vector: Tc_ext = [-20.741301 -94.464165 631.709309]

Rotation vector: omc_ext = [-2.641001 -1.310476 0.380756]

Rotation matrix: Rc_ext = [0.581524 0.756178 -0.300040
0.799226 -0.599880 0.037172
-0.151879 -0.261417 -0.953202]

Pixel error: err = [0.10622 0.07921]

Image 12:

Translation vector: Tc_ext = [107.021844 -35.444962 723.547178]

Rotation vector: omc_ext = [2.592209 -1.715766 -0.302986]

Rotation matrix: Rc_ext = [0.377686 -0.909999 -0.171040
-0.913542 -0.396338 0.091416
-0.150978 0.121725 -0.981014]

Pixel error: err = [0.14968 0.15417]

Image 13:

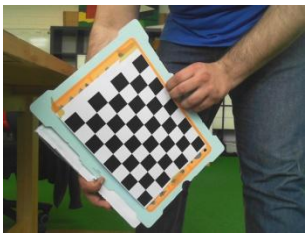
Translation vector: Tc_ext = [-29.231748 -119.886641 710.713031]

Rotation vector: omc_ext = [-2.044871 -0.787882 0.366839]

Rotation matrix: Rc_ext = [0.754274 0.392808 -0.526091
0.655458 -0.404115 0.638017
0.038017 -0.826070 -0.562283]

Pixel error: err = [0.33669 0.37194]

Image 14:



Translation vector: Tc_ext = [-18.273971 163.248209 680.885032]

Rotation vector: omc_ext = [-2.599560 0.962107 -0.519893]

Rotation matrix: Rc_ext = [0.706971 -0.554580 0.438900
-0.671037 -0.722000 0.168596
0.223386 -0.413711 -0.882577]

Pixel error: err = [0.25261 0.17799]

Image 15:

Translation vector: Tc_ext = [-47.534247 -128.071954 393.407013]

Rotation vector: omc_ext = [-2.068178 -2.009931 -0.103977]

Rotation matrix: Rc_ext = [0.043047 0.991172 -0.125398
0.972942 -0.013071 0.230680
0.227004 -0.131935 -0.964916]

Pixel error: err = [0.28796 0.33173]

Image 16:

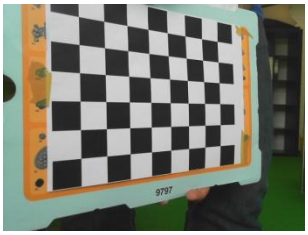
Translation vector: Tc_ext = [-114.837315 -104.026224 475.566251]

Rotation vector: omc_ext = [-1.962913 -1.845746 0.587670]

Rotation matrix: Rc_ext = [0.049168 0.838315 -0.542965
0.997921 -0.063916 -0.008317
-0.041676 -0.541427 -0.839714]

Pixel error: err = [0.65888 0.17351]

Image 17:



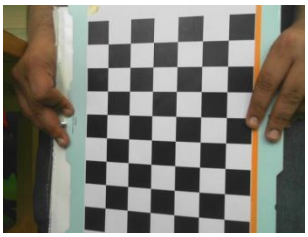
Translation vector: Tc_ext = [-80.526047 -104.284503 321.689315]

Rotation vector: omc_ext = [1.886443 1.849032 0.537912]

Rotation matrix: Rc_ext = [0.029301 0.827013 0.561418
0.999115 -0.007280 -0.041421
-0.030169 0.562135 -0.826495]

Pixel error: err = [0.63788 0.18015]

Image 18:



Translation vector: Tc_ext = [-62.566369 117.193227 390.045801]

Rotation vector: omc_ext = [-3.112575 -0.035757 0.344941]

Rotation matrix: Rc_ext = [0.975478 0.021619 -0.219032
0.023768 -0.999692 0.007183
-0.218809 -0.012213 -0.975691]

Pixel error: err = [0.20415 0.24769]

Image 19:

Translation vector: Tc_ext = [-133.819621 19.665625 367.184118]

Rotation vector: omc_ext = [-2.902128 -1.009990 0.452304]

Rotation matrix: Rc_ext = [0.746186 0.602294 -0.283632
0.612669 -0.787951 -0.061393
-0.260465 -0.127962 -0.956966]

Pixel error: err = [0.34990 0.21059]

Note: The numerical errors are approximately three times the standard deviations (for reference).

References:

[1] http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

[2] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 22(11):1330-1334, 2000.

[3] J.Y.Bouguet. MATLAB calibration tool. http://www.vision.caltech.edu/bouguetj/calib_doc/, Last updated Dec, 2nd, 2013

[4] Heikkilä, J. & Silvén, O., *A Four-step Camera Calibration Procedure with Implicit Image Correction*. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97), San Juan, Puerto Rico, p. 1106-1112, 1997

[5] Pollefeys, M. *The image of the absolute conic* http://www.cs.unc.edu/~marc/tutorial/node87.html, 2002