Minimal Autocalibration Pipeline

A minimal approach and experiments

Filippo Grotto VR460638
Matteo Meneghetti VR469054

Master degree in Computer Engineering for Robotics and Smart Industry

Computer Vision 2021/2022

Department of Computer Science
University of Verona

Contents

1 Problem Statement

The main idea of this project is to present a minimal pipeline to estimate intrinsic camera parameters using auto-calibration methods. We will assume known corresponding points and an initial estimation of the intrinsic camera parameters. The entire pipeline is based on a dataset provided by Zephyr, in particular, we used 3DFlow Dante dataset [5].

2 Pipeline

The pipeline is composed of two separate steps the first one to load the dataset and prepare a dedicated data-structure and the last one to perform the actual auto-calibration steps that will be discussed. The related files are *pipeline_dataset.m* and *pipeline_autocal.m*, respectively. Assuming the dataset provided by Zephyr with *Visibility.txt*. The only requirements for running those scripts are the *directory* variables defined on top of both.

2.1 Data Structure

Prepare the data structure from the Zephyr dataset. The idea is to build a node for each pair of images i, j with the related 3D point and the projected (uv, vv) in the images i and j called correspondence points. Moreover the related rotations R, translations t and the original intrinsic parameters K are collected starting from the *.xmp* files. In this way a symmetric cell $S\{i, j\}$ can be constructed. An example and instance for $S\{i, j\}$ is reported:

```
points: [2147×3 double]
uv_i: [2147×1 double]
vv_i: [2147×1 double]
uv_j: [2147×1 double]
vv_j: [2147×1 double]
name_view_i: '_SAM1001.JPG'
name_view_j: '_SAM1002.JPG'
K_i: [3×3 double]
R_i: [3×3 double]
t_i: [3×1 double]
K_j: [3×3 double]
R_j: [3×3 double]
```

```
t_j: [3×1 double]
```

Where *points* are the 3D points, *uu_i* and *vv_i* define a point on the image pixels i, *uv_j* and *vv_j* define a point on the image pixels j, *name_view_i* and *name_view_j* are the filenames of the related images and finally the parameters of the cameras for i and j.

2.2 Use centroids to select 3D points

Using the corresponding points of a pair of images it is possible to estimate the fundamental matrices. In order to make this estimation robust, only the 3D points close enough to the centroid are considered. The basic idea is the following:

$$||p - m|| < \epsilon \tag{1}$$

where p is the point, m is the mean of all points and $\epsilon = 3$ is the distance. Only the points close enough to the mean are selected. This selection is performed in the construction of the data-structure to reduce the number of edge cases in the computation.

2.3 Fundamental Matrices

The fundamental matrix is defined as:

$$m'^T F m = 0. \tag{2}$$

meaning that the corresponding point of m in the the second image should be in the epipolar line described by Fm. We can also observe that $det(F) = 0$ since $det([e']_x) = 0$ and that the definition of F is true even if we multiply by a scaling factor. The fundamental matrix is the relation of the pixel coordinates of the conjugate points. We can briefly mention this relation:

$$F = K'^{-T} E K^{-1} \tag{3}$$

The computation of the fundamental matrices can be performed using the pixel coordinates instead of normalized coordinates in the 8 points algorithm. The only property is the fact that F should be singular.

The problem of the 8 points linear algorithm is the sensitivity to noise and instability (ill-conditioned). Hartley proposed to use a standardization to improve the results [6]. This can be

performed by translating the points so that their origin matches and then scale them to get a mean distance of $\sqrt{2}$. Given T and T' the transformations of $\bar{m} = Tm$ and $\bar{m}' = T'm'$. If we use the points $\bar{m}$ and $\bar{m}'$ in the eight points algorithm we can compute $\bar{F}$ and recover the original fundamental matrix with $F = T'^T \bar{F} T$. Finally, the estimation of the linear algorithm can be refined using a geometric residual equal to the sum of the distances between points and conjugate epipolar lines.

$$min_F \sum_j d(Fm_i, m'_i)^2 + d(F^T m'_i m_i)^2 \tag{4}$$

where $d()$ is the distance point-line in the cartesian plane [8].

2.4 Epipolar lines

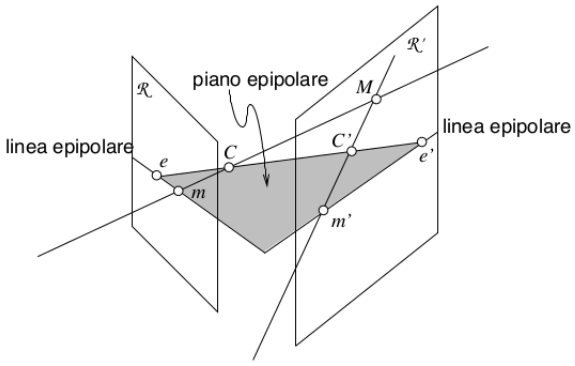

Figure 1: Epipolar geometry taken from [7]

The main ideas are collected in the Longuet-Higgins equation which is reported considering two projection perspective matrices $P = [Q|q]$ and $P' = [Q'|q']$:

$$0 = m'^T [e']_x Q' Q^{-1} m \tag{5}$$

which can be rewritten with the relation of the fundamental matrix:

$$m'^T F m = 0 \tag{6}$$

in fact if we consider m, the epipolar line is Fm. In Fig 2 and 3 some examples obtained drawing the epipolar lines in both directions are reported to validate the experimental results.

Figure 2: Epipolar lines of pair of images 1 and 2. It is visible the effect of the selection of 3D points

Figure 3: Epipolar lines of pair of images. It is visible the effect of the selection of 3D points

2.5 Autocalibration

Given the fundamental matrices the auto-calibration algorithms require an initial estimation of the matrix K_0 of internal camera parameters. A simple script is provided in *compute_k0.m* which extract the intrinsic camera parameters from the size of the image and the header information assuming a 35mm focal length. An example of initial estimation is reported:

$$K_0 = 1000 \begin{bmatrix} 5.9410 & 0 & 2.7360 \\ 0 & 5.9410 & 1.8240 \\ 0 & 0 & 0.0010 \end{bmatrix}$$

which is very close to the original values computed in Zephyr:

$$K = 1000 \begin{bmatrix} 5.7942 & 0 & 2.7923 \\ 0 & 5.7942 & 1.8174 \\ 0 & 0 & 0.0010 \end{bmatrix}$$

2.5.1 Medonca-Cipolla autocalibration

A first simple calibration method was proposed by Medonca and Cipolla in [2]. Considering the intrinsic parameters parametrized as

$$K = \begin{bmatrix} \alpha_x & s & u_0 \\ 0 & \epsilon\alpha_x & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

where α_x is the product of the focal length and the magnification factor, $\begin{bmatrix} u_0 & v_0 \end{bmatrix}^T$ are the coordinates of the principal point, s is the skew and ϵ is the aspect ratio. The main idea is to consider the following cost function for n images:

$$C(K_i, i = 1, \ldots, n) = \sum_{ij}^{n} \frac{w_{ij}(\sigma_{1ij} - \sigma_{2ij})}{\sum_{kl}^{n} w_{kl}\sigma_{2ij}} \tag{8}$$

where σ_{1ij} and σ_{2ij} are the non zero singular values of $K_j^T F_{ij} K_j$ in descending order, considering the fundamental matrix between two images i, j as F_{ij} and the intrinsic parameters K_j. The weights w_{ij} are the degree of confidence in the estimation of the fundamental matrix and can be equal to the number of points used in the computation of the fundamental matrix.

The implementation of this method is provided in *cost_medonca_cipolla.m* and can be used with $lsqnonlin$ or $fmincon$ to solve the related minimization problem. Several parameters has been tested but using the computed fundamental matrices and initial estimation the solver produces the following intrinsic parameters matrix with an error of 2.1854%:

$$K = 1000 \begin{bmatrix} 5.9209 & 0.0017 & 2.7613 \\ 0 & 5.8837 & 1.8216 \\ 0 & 0 & 0.0010 \end{bmatrix} \tag{9}$$

2.5.2 Kruppa method autocalibration

The second calibration method proposed is Kruppa method that can be found in [3] [9]. From the classical Kruppa's equations and cost function can be derived to be solved as a nonlinear least-squares optimization problem:

$$C(K_i, i = 1, \ldots, n) = \sum_{ij}^{n} \left\| \frac{F_{ij} w^{-1} F_{ij}^T}{||F_{ij} w^{-1} F_{ij}^T||} - \frac{[e_{ji}]_x w^{-1} [e_{ji}]^T}{||[e_{ji}]_x w^{-1} [e_{ji}]^T||} \right\| \tag{10}$$

where F_{ij} are the fundamental matrices, e are the epipoles and $w^{-1} = KK^T$. The norms used are the frobenius norms.

The implementation of this method is provided in *cost_kruppas_method.m* and can be used with $lsqnonlin$ or $fmincon$ to solve the related minimization problem. Several parameters has been tested but using the computed fundamental matrices and initial estimation the solver produces the following intrinsic parameters matrix with an error of 2.5332%:

$$K = 1000 \begin{bmatrix} 5.9410 & 0.0022 & 2.7361 \\ 0 & 5.9409 & 1.8256 \\ 0 & 0 & 0.0010 \end{bmatrix} \tag{11}$$

The method was tried as an experiment following the cost function reported in [9]. More advanced considerations can be made but they are out of the scope of this report.

2.5.3 Medonca-Cipolla toolbox

The last calibration method is provided in the Computer Vision Toolbox from Andrea Fusiello. The main idea is the following theorem from Huang and Faugeras [4]

$$det(E) = 0 \qquad \text{and} \qquad tr((EE^T))^2 - 2tr((EE^T)^2) = 0 \tag{12}$$

The previous equation is a condition to have one singular value equal to zero for E and two identical non-zero singular values. This condition can be used to build a cost function like:

$$C(K_i, i = 1, \ldots, n) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij}(tr((EE^T))^2 - 2tr((EE^T)^2))^2 \tag{13}$$

where $E_{ij} = K' F_{ij} K$ and $w_i j$ are the degree of confidence in the estimation of the fundamental matrix. The cost function is minimized with the matlab algorithms like the previous algorithms but the convergence is not guaranteed for each initial value. An example obtained with this procedure is reported with an error of 0.64652%:

$$K = 1000 \begin{bmatrix} 5.8317 & 0 & 2.7673 \\ 0 & 5.8317 & 1.8225 \\ 0 & 0 & 0.0010 \end{bmatrix} \tag{14}$$

2.5.4 Autocalibration considerations

As it is visible the auto-calibration algorithms improve the initial estimation based on the fundamental matrices provided. It is important to notice that both the initial estimation and the fundamental matrices play a big role on this estimation and its convergence. Another important considerations is related to the fact that the selecting points close to the mean is mandatory to have a robust estimation. The non-linear least square solver requires fine tuning to improve the accuracy of auto-calibration algorithms, in particular tolerances. Finally, the results obtained by the auto-calibration algorithms are good enough to reach reasonable results as will be presented in the following sections.

2.6 Relative orientation and 3D points evaluation

The relative orientation for each pair of images (i, j) is recovered using the estimated K using a linear algorithm and a non-linear refinement. The rotation R_{ji} and translation t_{ji} are saved in the related instance $S\{i, j\}$. To give an idea of the algorithm is to compute the normalized coordinates starting from pixel coordinates, compute the essential matrix using the eight point algorithm and finally use the decomposition of E to obtain the translation and rotation. Given $E = UDV^T$ the SVD of E it is possible to compute the following matrices

$$[t]_x = S = U(\pm S')U^T \qquad R = det(UV^T)UR'V^T \tag{15}$$

$$S' = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad R' = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{16}$$

Moreover a non-linear refinement can be found in [11]. Finally, the absolute orientation is recovered using the original 3D points and the triangulated 3D points from corresponding points. The orthogonal procrustes problem (opa) is solved to recover the scaling, the rotation and the translation as follow [10]:

$$D = s(RM + t) \tag{17}$$

To give an idea of the orthogonal procrustes analysis algorithm we have to consider the following three steps. The first step is about getting the value of R:

$$R = V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(VU^T) \end{bmatrix} U^T \qquad UDV^T = svd(\bar{y}\bar{x}^T) \tag{18}$$

where $\bar{x}$ and $\bar{y}$ are centralized coordinates of D and M. Secondly, it is necessary to compute the scale and the translation:

$$s = \frac{||\bar{y}||}{||\bar{x}||} \qquad t = \left(\frac{D}{s} - RM\right) \tag{19}$$

In this way we have computed all the necessary steps. Another possibility, without using the absolute orientation, is to concatenate the relative orientation of each view with respect to the first one, triangulate the points and register them with iterative closest point algorithm. However, the latter approach is not as robust as opa, it is computationally more expensive and it is not always possible to concatenate close views due to missing common points to compute the fundamental

matrices. In Fig 4 the results of relative orientations and orthogonal procrustes analysis are reported to consolidate the results obtained. In Fig 5 the final result is reported using the inliers points of 5 views and the estimated K.

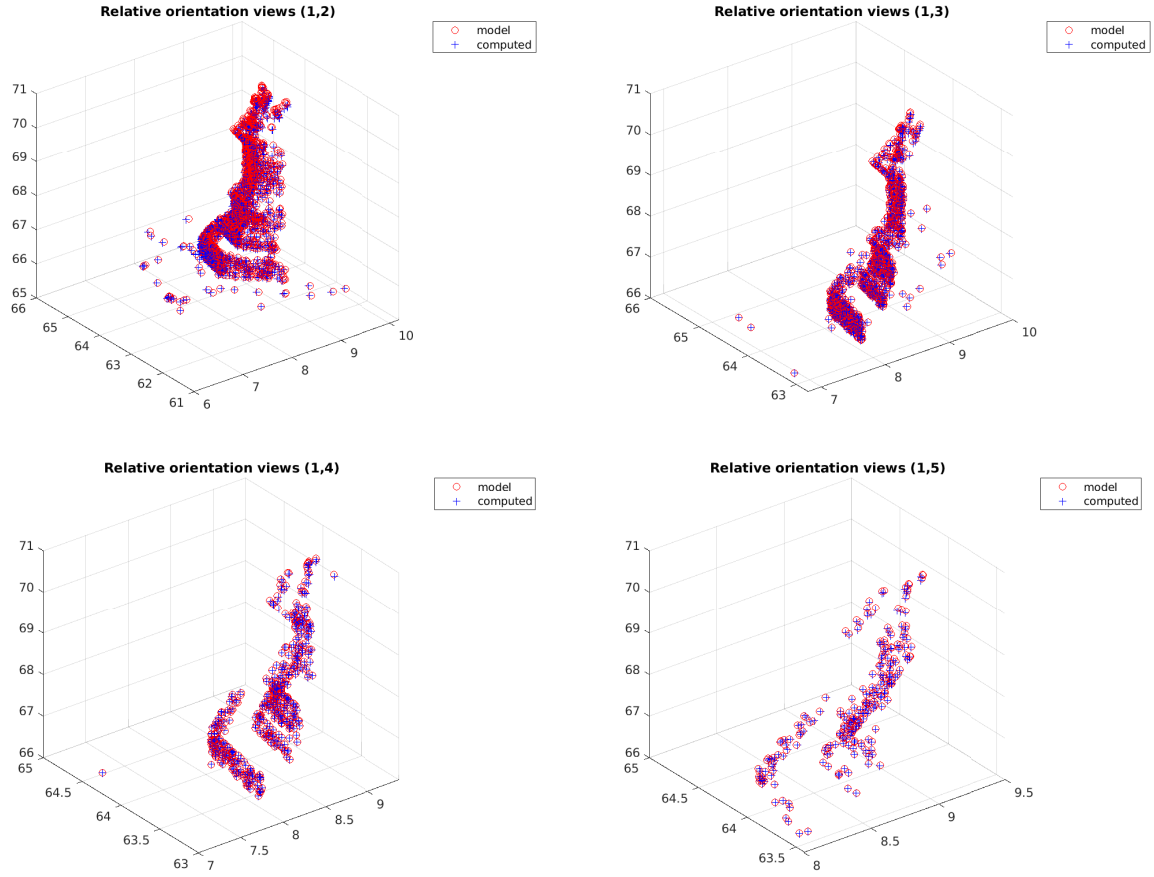

Figure 4: Example of relative orientations with OPA of the 3D points of the model and the computed ones. The considered views are reported in the images.

Dante Statue: Final result

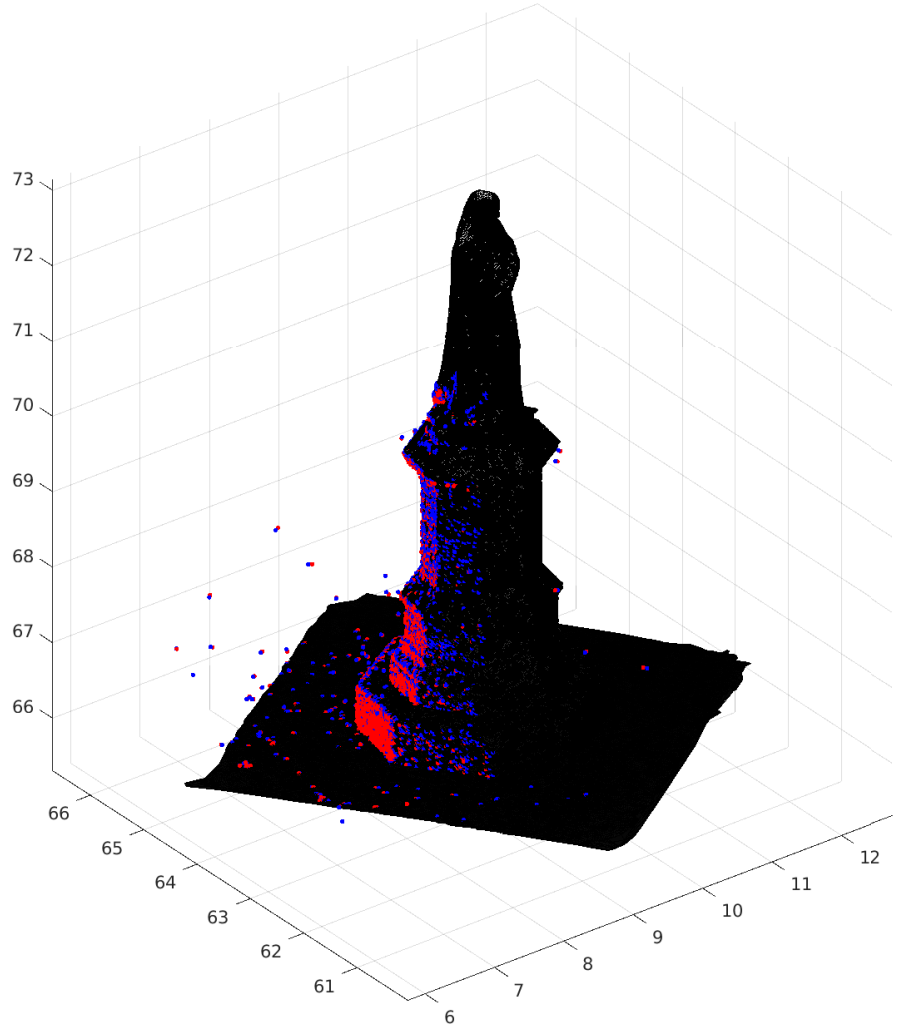

Figure 5: Final result using the 3D points obtained by 5 concatenated views. In blue the model and in red our projections. Our projections are not perfect but good enough to define the structure of the statue. This confirms the validity of the auto-calibration algorithms even though the computation time of the entire pipeline is not negligible (several minutes).

3 Final considerations

This project proposed a minimal pipeline to try auto-calibration algorithms. Three algorithms are presented and the problems related to the correct estimation of the fundamental matrices and the initial estimation has been briefly discussed. It is clear that a good initial estimation is necessary and ad-hoc consideration for the optimizations and the number of fundamental matrices are required. Other more advanced methods can be considered like the one proposed in [1] to make the entire process more robust and reliable.

4 Appendix A: Sample execution of the pipelines

The file *pipeline_dataset.m* will simply save the cell *S.mat* with the previously explained fields.

The file *pipeline_autocalibration.m* starting from the dataset and the cell previously computed will display the previously discussed images and some logs to show all the steps. An example is reported:

```
>> pipeline_autocal
Pipeline for autocalibration starting from S data-structure
Computing fundamental matrices
Fundamental nonlin Smps error views (1, 2):  0.14348
Fundamental nonlin Smps error views (1, 3):  0.14054
Fundamental nonlin Smps error views (1, 4):  0.1471
Fundamental nonlin Smps error views (1, 5):  0.14393
Fundamental nonlin Smps error views (1, 6):  0.14736
Fundamental nonlin Smps error views (1, 7):  0.14304
Fundamental nonlin Smps error views (2, 1):  0.14348
Fundamental nonlin Smps error views (2, 3):  0.14193
Fundamental nonlin Smps error views (2, 4):  0.14217
Fundamental nonlin Smps error views (2, 5):  0.14069
Fundamental nonlin Smps error views (2, 6):  0.14432
Fundamental nonlin Smps error views (2, 7):  0.14286
Fundamental nonlin Smps error views (3, 1):  0.14054
Fundamental nonlin Smps error views (3, 2):  0.14193
Fundamental nonlin Smps error views (3, 4):  0.13951
Fundamental nonlin Smps error views (3, 5):  0.14084
```

```
Fundamental nonlin Smps error views (3, 6):  0.1438
Fundamental nonlin Smps error views (3, 7):  0.14463
Fundamental nonlin Smps error views (4, 1):  0.1471
Fundamental nonlin Smps error views (4, 2):  0.14217
Fundamental nonlin Smps error views (4, 3):  0.13951
Fundamental nonlin Smps error views (4, 5):  0.1451
Fundamental nonlin Smps error views (4, 6):  0.14475
Fundamental nonlin Smps error views (4, 7):  0.1425
Fundamental nonlin Smps error views (4, 8):  0.14032
Fundamental nonlin Smps error views (4, 9):  0.13927
Fundamental nonlin Smps error views (5, 1):  0.14393
Fundamental nonlin Smps error views (5, 2):  0.14069
Fundamental nonlin Smps error views (5, 3):  0.14084
Fundamental nonlin Smps error views (5, 4):  0.1451
Fundamental nonlin Smps error views (5, 6):  0.14105
Fundamental nonlin Smps error views (5, 7):  0.13864
Fundamental nonlin Smps error views (5, 8):  0.14511
Fundamental nonlin Smps error views (5, 9):  0.13984
Fundamental nonlin Smps error views (5, 10):  0.14576
Fundamental nonlin Smps error views (6, 1):  0.14736
Fundamental nonlin Smps error views (6, 2):  0.14432
Fundamental nonlin Smps error views (6, 3):  0.1438
Fundamental nonlin Smps error views (6, 4):  0.14475
Fundamental nonlin Smps error views (6, 5):  0.14105
Fundamental nonlin Smps error views (6, 7):  0.14097
Fundamental nonlin Smps error views (6, 8):  0.1413
Fundamental nonlin Smps error views (6, 9):  0.1446
Fundamental nonlin Smps error views (6, 10):  0.14202
Fundamental nonlin Smps error views (7, 1):  0.14304
Fundamental nonlin Smps error views (7, 2):  0.14286
Fundamental nonlin Smps error views (7, 3):  0.14463
Fundamental nonlin Smps error views (7, 4):  0.1425
Fundamental nonlin Smps error views (7, 5):  0.13864
Fundamental nonlin Smps error views (7, 6):  0.14097
```

```
Fundamental nonlin Smps error views (7, 8):  0.14224
Fundamental nonlin Smps error views (7, 9):  0.13886
Fundamental nonlin Smps error views (7, 10):  0.14415
Fundamental nonlin Smps error views (8, 4):  0.14032
Fundamental nonlin Smps error views (8, 5):  0.14511
Fundamental nonlin Smps error views (8, 6):  0.1413
Fundamental nonlin Smps error views (8, 7):  0.14224
Fundamental nonlin Smps error views (8, 9):  0.14576
Fundamental nonlin Smps error views (8, 10):  0.14728
Fundamental nonlin Smps error views (9, 4):  0.13927
Fundamental nonlin Smps error views (9, 5):  0.13984
Fundamental nonlin Smps error views (9, 6):  0.1446
Fundamental nonlin Smps error views (9, 7):  0.13886
Fundamental nonlin Smps error views (9, 8):  0.14576
Fundamental nonlin Smps error views (9, 10):  0.14381
Fundamental nonlin Smps error views (10, 5):  0.14576
Fundamental nonlin Smps error views (10, 6):  0.14202
Fundamental nonlin Smps error views (10, 7):  0.14415
Fundamental nonlin Smps error views (10, 8):  0.14728
Fundamental nonlin Smps error views (10, 9):  0.14381
Estimating intrinsic parameters
Original
   1.0e+03 *

    5.7942         0    2.7923
         0    5.7942    1.8174
         0         0    0.0010

Initial estimation
   1.0e+03 *

    5.9410         0    2.7360
         0    5.9410    1.8240
         0         0    0.0010
```

```
Medonca cipolla:
   1.0e+03 *

    5.9209    0.0017    2.7613
         0    5.8837    1.8216
         0         0    0.0010

Autocalibration % error:  2.1854

Kruppas method:
   1.0e+03 *

    5.9410    0.0022    2.7361
         0    5.9409    1.8256
         0         0    0.0010

Autocalibration % error:  2.5332

Medonca cipolla (Toolbox):
   1.0e+03 *

    5.8317         0    2.7673
         0    5.8317    1.8225
         0         0    0.0010

Autocalibration % error:  0.64652

Computing relative orientations of views
Relative linear SO3 views (1, 2) error:  0.0056306
Relative nonlin SO3 views (1, 2) error:  0.0057026
Relative triang error views (1, 2):  0.027706
Relative linear SO3 views (1, 3) error:  0.012186
Relative nonlin SO3 views (1, 3) error:  0.012521
```

```
Relative triang error views (1, 3):  0.012344
Relative linear SO3 views (1, 4) error:  0.018683
Relative nonlin SO3 views (1, 4) error:  0.018827
Relative triang error views (1, 4):  0.0047162
Relative linear SO3 views (1, 5) error:  0.022338
Relative nonlin SO3 views (1, 5) error:  0.022124
Relative triang error views (1, 5):  0.0088446
Relative linear SO3 views (2, 3) error:  0.0075514
Relative nonlin SO3 views (2, 3) error:  0.0076158
Relative triang error views (2, 3):  0.032475
Relative linear SO3 views (2, 4) error:  0.014764
Relative nonlin SO3 views (2, 4) error:  0.014717
Relative triang error views (2, 4):  0.0073654
Relative linear SO3 views (2, 5) error:  0.019423
Relative nonlin SO3 views (2, 5) error:  0.019609
Relative triang error views (2, 5):  0.0063714
Relative linear SO3 views (3, 4) error:  0.0074843
Relative nonlin SO3 views (3, 4) error:  0.0074571
Relative triang error views (3, 4):  0.03073
Relative linear SO3 views (3, 5) error:  0.013501
Relative nonlin SO3 views (3, 5) error:  0.013511
Relative triang error views (3, 5):  0.010332
Relative linear SO3 views (4, 5) error:  0.0061366
Relative nonlin SO3 views (4, 5) error:  0.0060958
Relative triang error views (4, 5):  0.017958
Computing final points by view concatenation
Relative triang error views (1, 2):  0.027706
Relative triang error views (1, 3):  0.012344
Relative triang error views (1, 4):  0.0047162
Relative triang error views (1, 5):  0.0088446
Relative triang error views (2, 3):  0.032475
Relative triang error views (2, 4):  0.0073654
Relative triang error views (2, 5):  0.0063714
Relative triang error views (3, 4):  0.03073
```

```
Relative triang error views (3, 5):  0.010332
Relative triang error views (4, 5):  0.017958
Pipeline completed
```

References

[1] Riccardo Gherardi and Andrea Fusiello "Practical Autocalibration", ECCV10

[2] Medonca and Cipolla A Simple Technique for Self-Calibration

[3] A Study of Kruppa's Equation for Camera Self-calibration

[4] Huang T. and Faugeras O. Some properties of the E matrix in two-view motion estimation.

[5] Dante Dataset https://www.3dflow.net/3df-zephyr-reconstruction-showcase/

[6] Hartley R. I. (1995). In defence of the 8-point algorithm.

[7] Andrea Fusiello Visione computazionale. Tecniche di ricostruzione tridimensionale

[8] Luong Q.-T.; Faugeras O. D. (1996). The fundamental matrix: Theory, algorithms, and stability analysis.

[9] M. Pollefeys and L.V. Gool Stratified Self-Calibration with the Modulus Constraint

[10] Arun K. S.; Huang T. S.; Blostein S. D. (1987). Least-Squares Fitting of Two 3-D Point Sets.

[11] Horn B. (1990). Relative orientation.