A Software for Complete Calibration of Multicamera Systems

Tomáš Svoboda
with contributions from D. Martinec, T. Pajdla, O. Chum, T. Werner, and J. Bouguet

Czech Technical University, Faculty of Electrical Engineering
Center for Machine Perception, Prague, Czech Republic
http://cmp.felk.cvut.cz/~svoboda

Computer Vision Lab, ETH Zürich
Outline

- Motivation
- Problem definition
- Proposed solution
- Results
- Applications
Motivation

- multiple cameras became common
- they can be found in . . .
Virtual reality room
Telepresence setup
Calibration

- many tasks can be accomplished without knowing anything about the cameras
- however, many more when we know . . .
camera positions, and . . .
... camera orientations, and ...
camera internal parameters from geometry to pixels
nonlinear parameters included
Camera calibration is an old problem

- for photogrammetrists (even older problem)
- in computer vision
- many methods exist
Classical approaches — known 3D points
Classical approaches — plate at several positions

http://www.vision.caltech.edu/bouguetj/calib_doc/
Classical methods — revisited

Pros:

- many methods (and free codes)
- precise, even for complicated camera models
Classical methods — revisited

Pros:
- many methods (and free codes)
- precise, even for complicated camera models

Cons (for multicamera systems):
- many cameras $\rightarrow$ hand work is not an option
- large working volume to fill $\rightarrow$ big calibration objects/plates
Our solution — overview

We assume at least approximately synchronized multicamera ($N \geq 3$) setup.

- use 1-point calibration object easily detectable in images
- wave the calibration point through the working volume freely
- this will create a virtual calibration object (*but the 3D position unknown!*)
- apply theoretical results from self-calibration field
- estimate as complicated camera model as reasonable
- validate the results
Multiple cameras — Geometry

Problem definition:
From \( u^i_j \) points, for which \( \lambda^i_j u^i_j = P^i X_j \) holds
estimate Euclidean projection matrices \( P^i \)
and coordinates of the 3D points \( X_j \)
Pinhole camera model

\[
\lambda_j^i \begin{bmatrix}
  u_j^i \\
  v_j^i \\
  1
\end{bmatrix} = \lambda_j^i u_j^i = P^i X_j, \quad \lambda_j^i \in \mathcal{R}^+
\]

- \( j \) index points
- \( i \) index camera
- \( \lambda_j^i \) projective depths
- \( u_j^i \) point projections (we find them in images)
- \( X_j \) 3D points (we do not know the positions!)
- \( P^i \) camera matrices
Multicamera linear model

\[ W_s = \begin{bmatrix} \lambda_1^1 & \cdots & \lambda_1^n \\ \vdots & \ddots & \vdots \\ \lambda_m^1 & \cdots & \lambda_m^n \end{bmatrix} \begin{bmatrix} u_1^1 \\ v_1^1 \\ 1 \\ \vdots \\ u_n^m \\ v_n^m \\ 1 \end{bmatrix} = \begin{bmatrix} P^1 \\ \vdots \\ P^m \end{bmatrix} \begin{bmatrix} X_1 \cdots X_n \end{bmatrix}_{4 \times n} \]

Self-calibration (Euclidean stratification)

\[ W_s = PX = PH \hat{H}^{-1}X = \hat{P} \hat{X}, \]
What the software does:

1. Finds the projections $u_j$ of the laser pointer in the images.
What the software does:

1. Finds the projections $u^i_j$ of the laser pointer in the images.

2. Discards misdetected points by pairwise RANSAC analysis.
What the software does:

1. Finds the projections $u^i_j$ of the laser pointer in the images.

2. Discards misdetected points by pairwise RANSAC analysis.

3. Estimates projective depths $\lambda^i_j$ and fills the missing points to make scaled measurement matrix $W_s$ complete.
What the software does:

1. Finds the projections $\mathbf{u}_j^i$ of the laser pointer in the images.

2. Discards misdetected points by pairwise RANSAC analysis.

3. Estimates projective depths $\lambda^i_j$ and fills the missing points to make scaled measurement matrix $W_s$ complete.

4. Performs the rank 4 factorization of the matrix $W_s$ to get projective shape and motion and upgrades them to Euclidean ones.
What the software does:

1. Finds the projections $u^i_j$ of the laser pointer in the images.

2. Discards misdetected points by pairwise RANSAC analysis.

3. Estimates projective depths $\lambda^i_j$ and fills the missing points to make scaled measurement matrix $W_s$ complete.

4. Performs the rank 4 factorization of the matrix $W_s$ to get projective shape and motion and upgrades them to Euclidean ones.

5. Estimates the parameters of the non-linear distortion
What the software does:

1. Finds the projections $u^i_j$ of the laser pointer in the images.

2. Discards misdetected points by pairwise RANSAC analysis.

3. Estimates projective depths $\lambda^i_j$ and fills the missing points to make scaled measurement matrix $W_s$ complete.

4. Performs the rank 4 factorization of the matrix $W_s$ to get projective shape and motion and upgrades them to Euclidean ones.

5. Estimates the parameters of the non-linear distortion

6. Optionally, if some true 3D information is known, aligns the computed Euclidean structures with a world system.
What the software does:

1. Finds the projections $u_{ij}^i$ of the laser pointer in the images.
2. Discards misdetected points by pairwise RANSAC analysis.
3. Estimates projective depths $\lambda_{ij}^i$ and fills the missing points to make scaled measurement matrix $W_s$ complete.
4. Performs the rank 4 factorization of the matrix $W_s$ to get projective shape and motion and upgrades them to Euclidean ones.
5. Estimates the parameters of the non-linear distortion
6. Optionally, if some true 3D information is known, aligns the computed Euclidean structures with a world system.

Many cross-validation steps inside.
**Calibration object**

A very standard laser pointer with a piece of transparent plastic attached.
Finding points

Needs to be a bit more clever than a simple thresholding

Statistical analysis of the images (almost) solves it.
Finding points

Sub-pixel accuracy is desirable

![Graphs and images illustrating active ROI, PSF approximation by 2D Gaussian, correlation coefficients, and interpolated ROI.]

Around 100 ms per image.
Calibration input

Video
We know

\[ W_s = \begin{bmatrix}
\lambda_1^1 & u_1^1 & v_1^1 \\
\vdots & \vdots & \vdots \\
\lambda_m^1 & u_1^m & v_1^m \\
\lambda_1^n & u_n^1 & v_n^1 \\
\vdots & \vdots & \vdots \\
\lambda_m^n & u_n^m & v_n^m
\end{bmatrix}
\cdots
\begin{bmatrix}
\lambda_1^1 & u_1^1 & v_1^1 \\
\vdots & \vdots & \vdots \\
\lambda_m^1 & u_n^1 & v_n^1 \\
\lambda_1^n & u_n^m & v_n^m \\
\vdots & \vdots & \vdots \\
\lambda_m^n & u_n^m & v_n^m
\end{bmatrix}
\begin{bmatrix}
P_1 \\
\vdots \\
P_m
\end{bmatrix}
= \begin{bmatrix}
P_1 \\
\vdots \\
P_m
\end{bmatrix}
[\mathbf{X}_1 \cdots \mathbf{X}_n]_{4 \times n}
\]

\[ W_s = \mathbf{P}\mathbf{X} = \mathbf{P}\mathbf{H}\mathbf{H}^{-1}\mathbf{X} = \hat{\mathbf{P}}\hat{\mathbf{X}}, \]

However, some \([u_j^i, v_j^i]^\top\) may be missing!
Estimation of $\lambda_{j}^{i}$
(Sturm & Triggs ECCV96)

uses the epipolar geometry

$$
\lambda_{j}^{i} = \frac{(e^{ik} \times u_{j}^{i}) \cdot (F^{ik} u_{j}^{k})}{\|e^{ik} \times u_{j}^{i}\|^2} \lambda_{j}^{k}
$$
We know

\[
W_s = \begin{bmatrix}
\lambda_1^1 & u_1^1 & v_1^1 & 1 \\
\vdots & \vdots & \vdots & \vdots \\
\lambda_n^1 & u_n^1 & v_n^1 & 1 \\
\lambda_1^m & u_1^m & v_1^m & 1 \\
\vdots & \vdots & \vdots & \vdots \\
\lambda_n^m & u_n^m & v_n^m & 1
\end{bmatrix} = \begin{bmatrix}
P^1 \\
\vdots \\
P^m
\end{bmatrix}_{3m \times 4} \begin{bmatrix}
X_1 \\
\vdots \\
X_n
\end{bmatrix}_{4 \times n}
\]

However, some \([u_j^i, v_j^i]^{\top}\) and \(\lambda_j^i\) may be missing!
Filling missing points
(Martinec and Pajdla ECCV2002)

Example: \[ R = \begin{bmatrix} 4 & 6 \\ 2 & \times \\ \times & 3 \end{bmatrix}, \text{for rank } R = 1 \text{ instead of rank } R = 4 \]

\[ B_1 = \text{Span}( \begin{bmatrix} 4 & 0 \\ 2 & 0 \\ 0 & 1 \end{bmatrix} ), \]

\[ B_2 = \text{Span}( \begin{bmatrix} 6 & 0 \\ 0 & 1 \\ 3 & 0 \end{bmatrix} ) \]

all possible fillings

...linear hull \( B_1 \)

\[ B \subseteq B_1 \cap B_2 = \text{Span}( \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} ) \]

\[ \tilde{R} = \begin{bmatrix} 4 & 6 \\ 2 & \tilde{x} \\ 2 & 3 \end{bmatrix} \]
We know

\[
W_s = \begin{bmatrix}
\lambda_1^1 & \begin{bmatrix} u_1^1 & v_1^1 \\ 1 & 1 \end{bmatrix} & \cdots & \lambda_n^1 & \begin{bmatrix} u_n^1 & v_n^1 \\ 1 & 1 \end{bmatrix} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\lambda_1^m & \begin{bmatrix} u_1^m & v_1^m \\ 1 & 1 \end{bmatrix} & \cdots & \lambda_n^m & \begin{bmatrix} u_n^m & v_n^m \\ 1 & 1 \end{bmatrix}
\end{bmatrix}

= \begin{bmatrix}
P_1 \\
p_2 \\
p_3 \\
p_m
\end{bmatrix}_{3m \times 4} [X_1 \cdots X_n]_{4 \times n}
\]

\[
W_s = PX = PHH^{-1}X = \hat{P}\hat{X},
\]
Rank–4 factorization

\[ W_s = \begin{bmatrix} P_1 \\ \vdots \\ P_m \end{bmatrix}_{3m \times 4} [X_1 \cdots X_n]_{4 \times n} \]

So, matrix \( W_s \) should have rank at most 4

\[ W_s = USV^\top \]

\[ \begin{bmatrix} P_1 \\ \vdots \\ P_m \end{bmatrix}_{3m \times 4} [X_1 \cdots X_n]_{4 \times n} = (U\sqrt{S_4})(\sqrt{S_4}V^\top) \]

where \( S_4 \) is the \( S \) with only 4 biggest diagonal values, rest is zeroed.
We know

\[ W_s = PX = PHH^{-1}X = \hat{P}\hat{X}, \]

We must find a $4 \times 4$ matrix $H$ which upgrades the projective structures $P, X$ to metric ones, $\hat{P}, \hat{X}$. 
Euclidean stratification
(Pollefeys et al, Hartley, . . . )

based on the idea of absolute quadric (conic)

\[ \hat{P}^i = \mu_i \left[ K^i R^i \ K^i t^i \right] \]

\[ \hat{P}^i \hat{\Omega}_\infty \hat{P}^{i\top} \sim K^i K^{i\top} \]

where

\[ \hat{\Omega}_\infty = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix} \]
Euclidean stratification cont.

absolute conic exists also in the projective world!

\[ K^i K^{i\top} \sim (\hat{P}^i H^{-1})(H\hat{\Omega}_\infty H^{\top})(H^{-\top}\hat{P}^i)^{\top} \]

\[ K^i K^{i\top} \sim P^i \Omega_\infty P^{i\top} \]

We know the projective \( P^i \). The projective \( \Omega_\infty \) is \( 4 \times 4 \) symmetric.

Once \( \Omega_\infty \) is known, then we can compute \( H \) from

\[ \Omega_\infty = H\hat{\Omega}_\infty H^{\top} \]

by eigenvalue decomposition and get the sought Euclidean structures \( \hat{P}^i = P^i H \) and \( \hat{X}_j = H^{-1}X_j \).
Euclidean stratification — Example of solution

assume everything is known except focal lengths

\[
K^i = \begin{bmatrix}
  f^i & 0 & u_0^i \\
  0 & \alpha_i f^i & v_0^i \\
  0 & 0 & 1
\end{bmatrix} \rightarrow K^i K^{i\top} = \begin{bmatrix}
  f^{i2} & 0 & 0 \\
  0 & f^{i2} & 0 \\
  0 & 0 & 1
\end{bmatrix}
\]

Remember that \( K^i K^{i\top} \sim P^i \Omega_\infty P^{i\top} \)

\[
(P^i \Omega_\infty P^{i\top})_{11} - (P^i \Omega_\infty P^{i\top})_{22} = 0
\]

\[
(P^i \Omega_\infty P^{i\top})_{12} = 0
\]

\[
(P^i \Omega_\infty P^{i\top})_{13} = 0
\]

\[
(P^i \Omega_\infty P^{i\top})_{23} = 0
\]

Each camera contributes by 4 contraints.
We have the metric linear model

\[ W_s = \hat{P}\hat{X} \]

Estimation of non-linear distortion starts from

\[ \hat{X}_j \leftrightarrow u^i_j \]

correspondences. We use the CalTech package

http://www.vision.caltech.edu/bouguetj/calib_doc/

Then it goes back, adapt parameters and . . .
Aligning the results with the world

User provides some 3D information. Example: “Cameras No. 11,13,15 define the $xy$ plane”.

reconstructed points/camera setup only inliers are used

Graphical Output Validation: View from the top camera
The calibration “point” needs not to be visible in all cameras!
Results — Calibrated setups

Graphical Output Validation: View from the top camera

Graphical Output Validation: Aligned data
Results — Linear model

measured, o, vs reprojected, +, 2D points (camera: 12)

2D error: mean (blue), std (red)
Results — Complete model

Very fine results from (almost) nothing!
Results — Simple setup

reconstructed points/camera setup only inliers are used

2D error: mean (blue), std (red)
Application example — volumetric reconstruction

I know, it is just toy example. Still, it shows that the metric is OK.
Application example — mobile multicamera setup

Video
Mobile multicamera setup - worker 3D tracking
Mobile multicamera setup - worker 3D tracking
Summary

- waving the point object is the only hand work required
- no user interaction
- complete calibration of 16 camera setup may be done in 60-90 minutes (95% computation)

Codes, sample data, papers, etc. downloadable from http://cmp.felk.cvut.cz/~svoboda/SelfCal
Problem definition:
From $u_j$ points, for which $\lambda_j u_j = P^i X_j$ holds
estimate Euclidean projection matrices $P^i$ and coordinates of the 3D points $X_j$
Example: \( R = \begin{bmatrix} 4 & 6 \\ 2 & \times \\ \times & 3 \end{bmatrix} \), for rank \( R = 1 \) instead of rank \( R = 4 \)

\[ \begin{bmatrix} 4 \\ 2 \\ \times \end{bmatrix} \quad \text{all possible fillings} \]

\[ \ldots \quad B_1 = \text{Span}( \begin{bmatrix} 4 & 0 \\ 2 & 0 \\ 0 & 1 \end{bmatrix} ) \quad \text{linear hull } B_1 \]

\[ \begin{bmatrix} 6 \\ \times \\ 3 \end{bmatrix} \quad \ldots \quad B_2 = \text{Span}( \begin{bmatrix} 6 & 0 \\ 0 & 1 \\ 3 & 0 \end{bmatrix} ) \]

\[ B \subseteq B_1 \cap B_2 = \text{Span}( \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} ) \]

\( \tilde{R} = \begin{bmatrix} 4 & 6 \\ 2 & 3 \\ 2 & 3 \end{bmatrix} \)
reconstructed points/camera setup only inliers are used
Graphical Output Validation: View from the top camera
measured, o, vs reprojected, +, 2D points (camera: 4)
measured, o, vs reprojected, +, 2D points (camera: 40)
Graphical Output Validation: View from the top camera
measured, o, vs reprojected, +, 2D points (camera: 12)
2D error: mean (blue), std (red)

Id of the camera

pixels

Id of the camera

0 2 4 6 8 10 12 14 16 18
measured, o, vs reprojected, +, 2D points (camera: 12)
2D error: mean (blue), std (red)
reconstructed points/camera setup only inliers are used