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

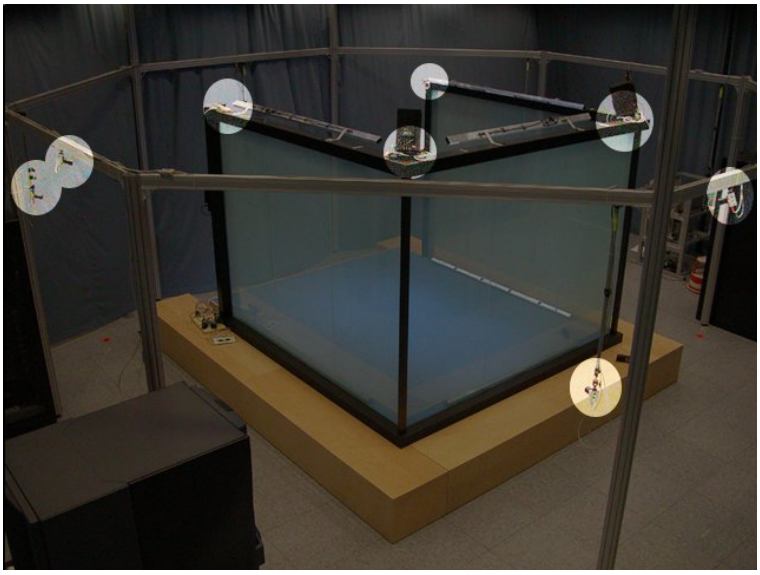
Outline

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

Motivation

- multiple cameras became common
- they can be found in . . .

Virtual reality room

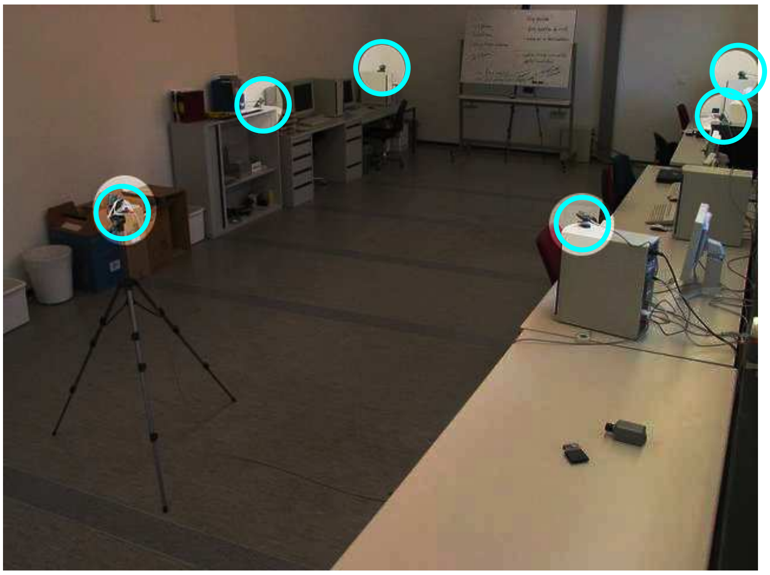


Telepresence setup

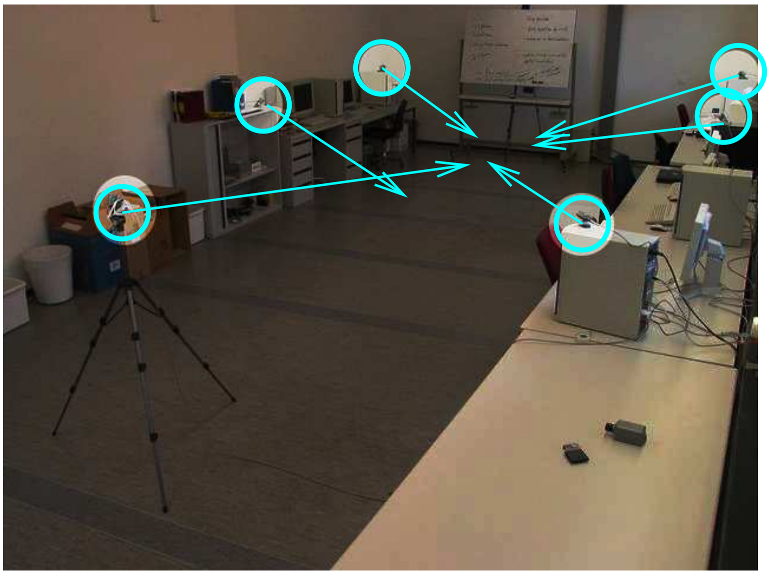
Calibration

- many tasks can be accomplished without knowing anything about the cameras
- however, many more when we know . . .

camera positions, and . . .

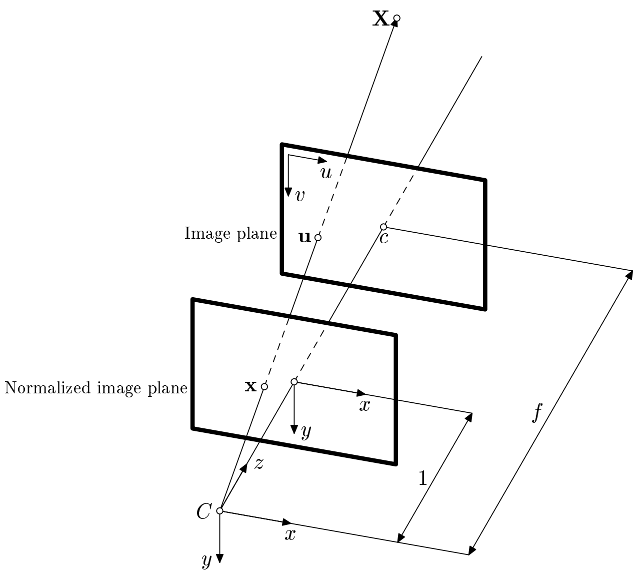


. . . camera orientations, and . . .



. . . camera internal parameters

from geometry to pixels

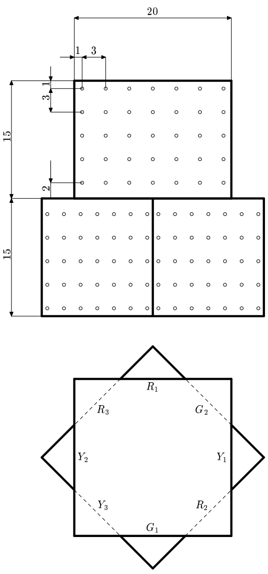


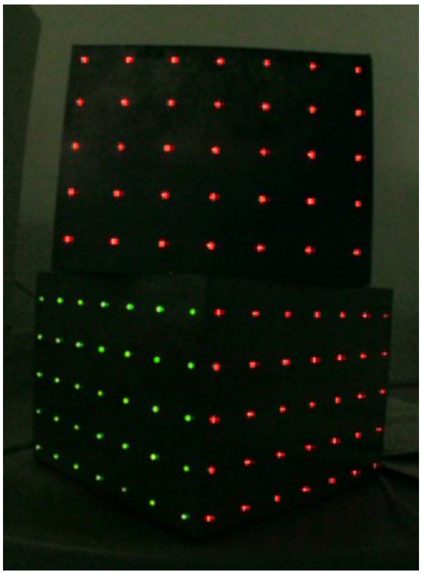
. . . nonlinear parameters included

Camera calibration is an old problem

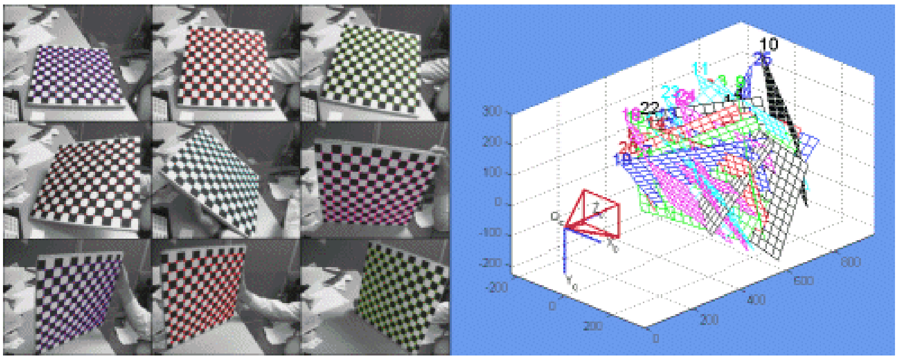
- for photogrammetrists (even older problem)
- in computer vision
- many methods exist

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

Cons (for multicamera systems):

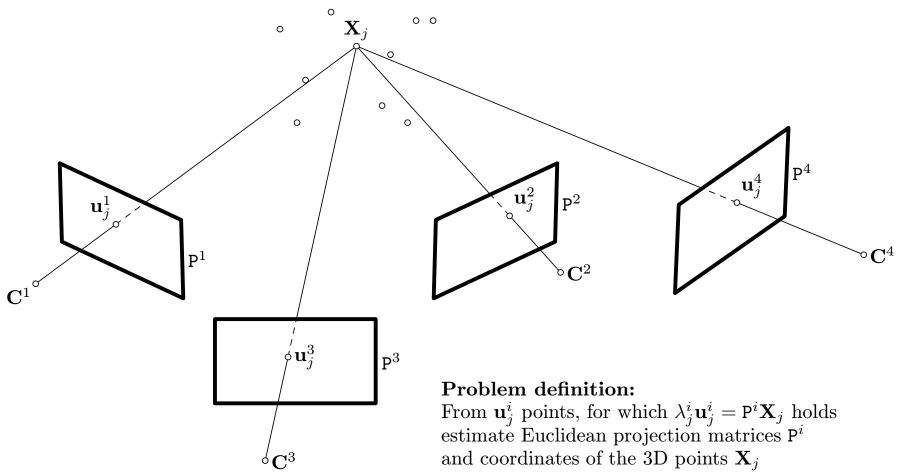
- many cameras → hand work is not an option
- large working volume to fill → big calibration objects/plates

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 $\mathbf{u}_j^i$ points, for which $\lambda_j^i \mathbf{u}_j^i = \mathtt{P}^i \mathbf{X}_j$ holds
estimate Euclidean projection matrices $\mathtt{P}^i$
and coordinates of the 3D points $\mathbf{X}_j$

Pinhole camera model

$$\lambda_j^i \begin{bmatrix} u_j^i \\ v_j^i \\ 1 \end{bmatrix} = \lambda_j^i \mathbf{u}_j^i = \mathtt{P}^i \mathbf{X}_j \,, \qquad \lambda_j^i \in \mathcal{R}^+$$

- $_j$ index points
- ^{i} index camera
- λ_j^i projective depths
- $\mathbf{u}_j^i$ point projections (we find them in images)
- $\mathbf{X}_j$ 3D points (we do not know the positions!)
- $\mathtt{P}^i$ camera matrices

Multicamera linear model

$$\mathtt{W}_s = \begin{bmatrix} \lambda_1^1 \begin{bmatrix} u_1^1 \\ v_1^1 \\ 1 \end{bmatrix} & \cdots & \lambda_n^1 \begin{bmatrix} u_n^1 \\ v_n^1 \\ 1 \end{bmatrix} \\ \vdots & \vdots & \vdots \\ \lambda_1^m \begin{bmatrix} u_1^m \\ v_1^m \\ 1 \end{bmatrix} & \cdots & \lambda_n^m \begin{bmatrix} u_n^m \\ v_n^m \\ 1 \end{bmatrix} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathtt{P}^1 \\ \vdots \\ \mathtt{P}^m \end{bmatrix}_{3m\times 4}}_{\mathtt{P}} \underbrace{[\mathbf{X}_1 \cdots \mathbf{X}_n]_{4\times n}}_{\mathtt{X}}$$

Self-calibration (Euclidean stratification)

$$\mathtt{W}_s = \mathtt{P}\mathtt{X} = \underbrace{\mathtt{P}\mathtt{H}}\,\underbrace{\mathtt{H}^{-1}\mathtt{X}} = \hat{\mathtt{P}}\hat{\mathtt{X}} \,,$$

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 λ_j^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.

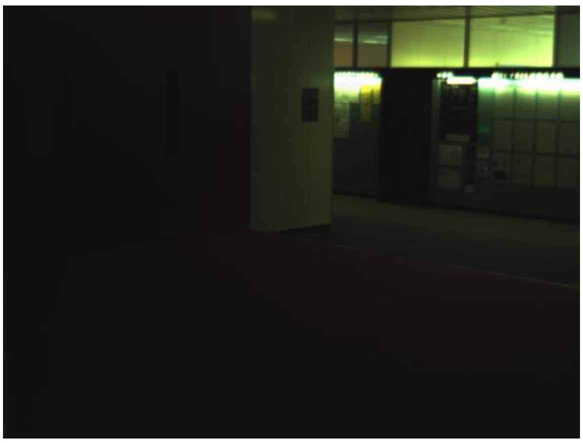
Many cross-validation steps inside.

Calibration object


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

Finding points

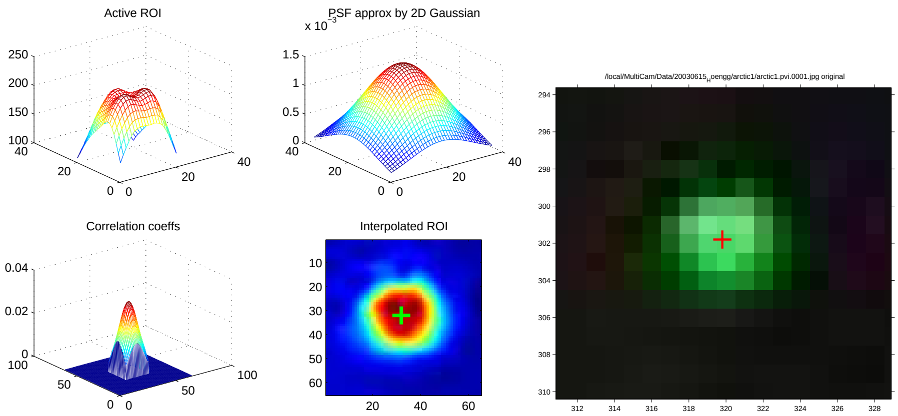
Needs to be a bit more clever than a simple thresholding



Statistical analysis of the images (almost) solves it.

Finding points

Sub-pixel accuracy is desirable



Around 100 ms per image.

Calibration input

Video

We know

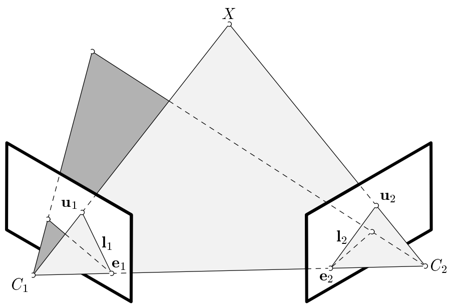
$$\mathtt{W}_s = \begin{bmatrix} \lambda_1^1 \begin{bmatrix} u_1^1 \\ v_1^1 \\ 1 \end{bmatrix} & \cdots & \lambda_n^1 \begin{bmatrix} u_n^1 \\ v_n^1 \\ 1 \end{bmatrix} \\ \vdots & \vdots & \vdots \\ \lambda_1^m \begin{bmatrix} u_1^m \\ v_1^m \\ 1 \end{bmatrix} & \cdots & \lambda_n^m \begin{bmatrix} u_n^m \\ v_n^m \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \mathtt{P}^1 \\ \vdots \\ \mathtt{P}^m \end{bmatrix}_{3m \times 4} [\mathbf{X}_1 \cdots \mathbf{X}_n]_{4 \times n}$$

$$\mathtt{W}_s = \mathtt{P}\mathtt{X} = \mathtt{P}\mathtt{H}\mathtt{H}^{-1}\mathtt{X} = \hat{\mathtt{P}}\hat{\mathtt{X}} \,,$$

However, some $[u_j^i, v_j^i]^\top$ may be missing!

Estimation of λ_j^i (Sturm & Triggs ECCV96)

uses the epipolar geometry



$$\lambda_j^i = \frac{(\mathbf{e}^{ik} \times \mathbf{u}_j^i) \cdot (\mathtt{F}^{ik}\mathbf{u}_j^k)}{\|\mathbf{e}^{ik} \times \mathbf{u}_j^i\|^2}\lambda_j^k$$

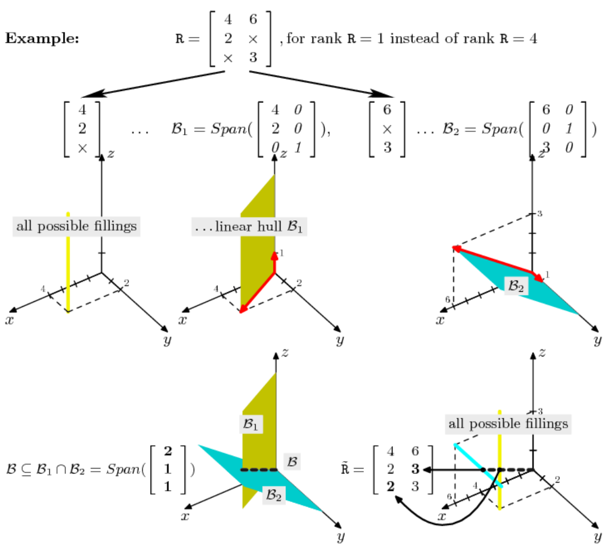
We know

$$\mathtt{W}_s = \begin{bmatrix} \lambda_1^1 \begin{bmatrix} u_1^1 \\ v_1^1 \\ 1 \end{bmatrix} & \cdots & \lambda_n^1 \begin{bmatrix} u_n^1 \\ v_n^1 \\ 1 \end{bmatrix} \\ \vdots & \vdots & \vdots \\ \lambda_1^m \begin{bmatrix} u_1^m \\ v_1^m \\ 1 \end{bmatrix} & \cdots & \lambda_n^m \begin{bmatrix} u_n^m \\ v_n^m \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \mathtt{P}^1 \\ \vdots \\ \mathtt{P}^m \end{bmatrix}_{3m\times 4} [\mathbf{X}_1 \cdots \mathbf{X}_n]_{4\times n}$$

$$\mathtt{W}_s = \mathtt{PX} = \mathtt{PHH}^{-1}\mathtt{X} = \hat{\mathtt{P}}\hat{\mathtt{X}}\,,$$

However, some $[u_j^i, v_j^i]^\top$ and λ_j^i may be missing!

Filling missing points (Martinec and Pajdla ECCV2002)



We know

$$\mathtt{W}_s = \begin{bmatrix} \lambda_1^1 \begin{bmatrix} u_1^1 \\ v_1^1 \\ 1 \end{bmatrix} & \cdots & \lambda_n^1 \begin{bmatrix} u_n^1 \\ v_n^1 \\ 1 \end{bmatrix} \\ \vdots & \vdots & \vdots \\ \lambda_1^m \begin{bmatrix} u_1^m \\ v_1^m \\ 1 \end{bmatrix} & \cdots & \lambda_n^m \begin{bmatrix} u_n^m \\ v_n^m \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \mathtt{P}^1 \\ \vdots \\ \mathtt{P}^m \end{bmatrix}_{3m\times 4} [\mathbf{X}_1 \cdots \mathbf{X}_n]_{4\times n}$$

$$\mathtt{W}_s = \mathtt{PX} = \mathtt{PHH}^{-1}\mathtt{X} = \hat{\mathtt{P}}\hat{\mathtt{X}} \,,$$

Rank–4 factorization

$$\mathtt{W}_s = \begin{bmatrix} \mathtt{P}^1 \\ \vdots \\ \mathtt{P}^m \end{bmatrix}_{3m\times \mathbf{4}} [\mathbf{X}_1 \cdots \mathbf{X}_n]_{\mathbf{4}\times n}$$

So, matrix $\mathtt{W}_s$ should have rank at most 4

$$\mathtt{W}_s = \mathtt{USV}^\top$$

$$\begin{bmatrix} \mathtt{P}^1 \\ \vdots \\ \mathtt{P}^m \end{bmatrix}_{3m\times 4} [\mathbf{X}_1 \cdots \mathbf{X}_n]_{4\times n} = (\mathtt{U}\sqrt{\mathtt{S}_4})(\sqrt{\mathtt{S}_4}\mathtt{V}^\top)$$

where $\mathtt{S}_4$ is the $\mathtt{S}$ with only 4 biggest diagonal values, rest is zeroed.

We know

$$\mathtt{W}_s = \mathtt{PX} = \mathtt{PHH}^{-1}\mathtt{X} = \hat{\mathtt{P}}\hat{\mathtt{X}} \,,$$

We must find a 4×4 matrix $\mathtt{H}$ which upgrades the projective structures $\mathtt{P}, \mathtt{X}$ to metric ones, $\hat{\mathtt{P}}, \hat{\mathtt{X}}$.

Euclidean stratification (Pollefeys et al, Hartley, . . .)

based on the idea of absolute quadric (conic)

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

$$\hat{\mathtt{P}}^i \hat{\Omega}_\infty \hat{\mathtt{P}}^{i\top} \sim \mathtt{K}^i\mathtt{K}^{i\top}$$

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!

$$\begin{aligned} \mathtt{K}^i\mathtt{K}^{i\top} &\sim (\hat{\mathtt{P}}^i\mathtt{H}^{-1})(\mathtt{H}\hat{\Omega}_\infty\mathtt{H}^\top)(\mathtt{H}^{-\top}\hat{\mathtt{P}}^{i\top}) \\ \mathtt{K}^i\mathtt{K}^{i\top} &\sim \mathtt{P}^i\Omega_\infty\mathtt{P}^{i\top} \end{aligned}$$

We know the projective $\mathtt{P}^i$. The projective Ω_∞ is 4×4 symmetric.

Once Ω_∞ is known, then we can compute $\mathtt{H}$ from

$$\Omega_\infty = \mathtt{H}\hat{\Omega}_\infty\mathtt{H}^\top$$

by eigenvalue decomposition and get the sought Euclidean structures $\hat{\mathtt{P}}^i = \mathtt{P}^i\mathtt{H}$ and $\hat{\mathbf{X}}_j = \mathtt{H}^{-1}\mathbf{X}_j$.

Euclidean stratification — Example of solution

assume everything is known except focal lenghts

$$\mathtt{K}^i = \begin{bmatrix} f^i & 0 & u_0^i \\ 0 & \alpha^i f^i & v_0^i \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \mathtt{K}^i\mathtt{K}^{i\top} = \begin{bmatrix} f^{i2} & 0 & 0 \\ 0 & f^{i2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Remember that $\mathtt{K}^i\mathtt{K}^{i\top} \sim \mathtt{P}^i\Omega_\infty\mathtt{P}^{i\top}$

$$\begin{aligned} (\mathtt{P}^i\Omega_\infty\mathtt{P}^{i\top})_{11} - (\mathtt{P}^i\Omega_\infty\mathtt{P}^{i\top})_{22} &= 0 \\ (\mathtt{P}^i\Omega_\infty\mathtt{P}^{i\top})_{12} &= 0 \\ (\mathtt{P}^i\Omega_\infty\mathtt{P}^{i\top})_{13} &= 0 \\ (\mathtt{P}^i\Omega_\infty\mathtt{P}^{i\top})_{23} &= 0 \end{aligned}$$

Each camera contributes by 4 contraints.

We have the metric linear model

$$\mathtt{W}_s = \hat{\mathtt{P}}\hat{\mathtt{X}}$$

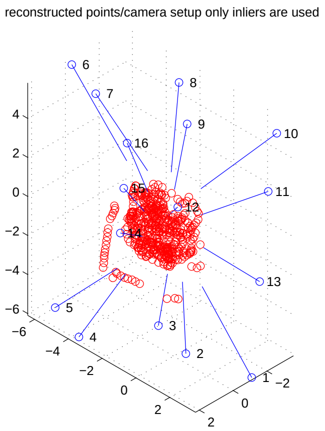
Estimation of non-linear distortion starts from

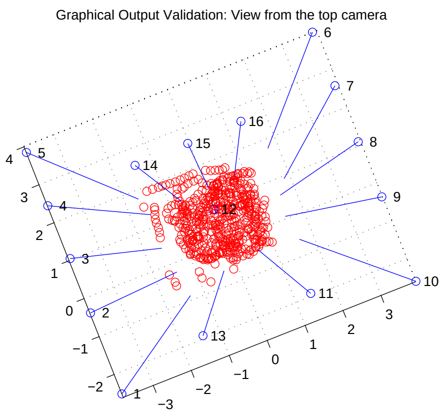
$$\hat{\mathtt{X}}_j \leftrightarrow \mathbf{u}_j^i$$

correspondences. We use the CalTech package
http://www.vision.caltech.edu/bouguetj/calib_doc/

Then it goes back, adapt parameters and . . .

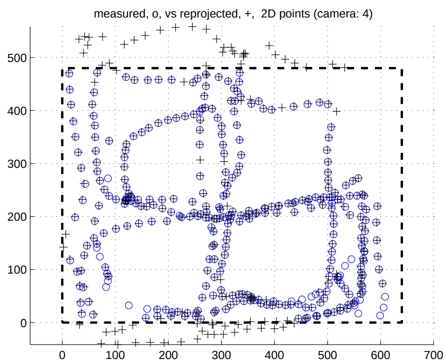
Aligning the results with the world

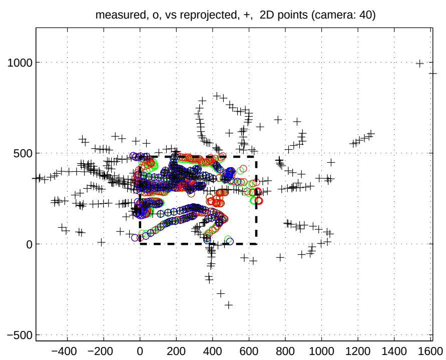




User provides some 3D information. Example: "Cameras No. 11,13,15 define the xy plane".

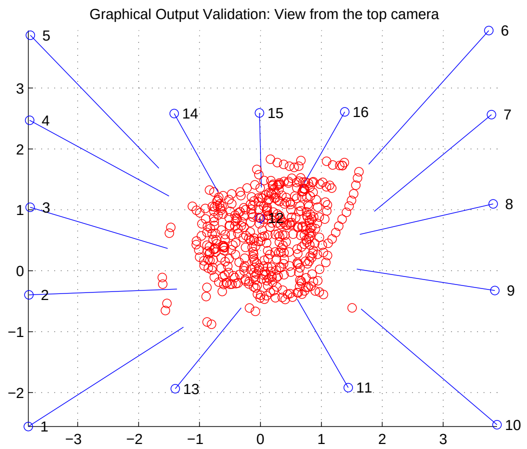
Results — Filling points

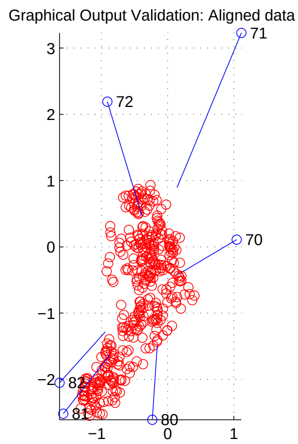




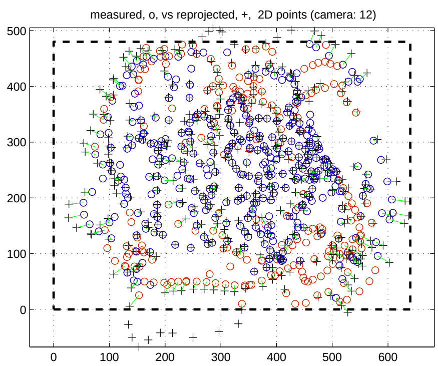
The calibration "point" needs not to be visible in all cameras!

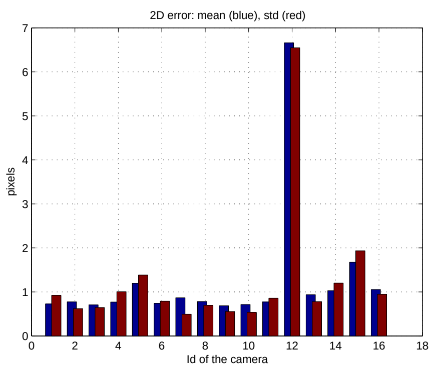
Results — Calibrated setups



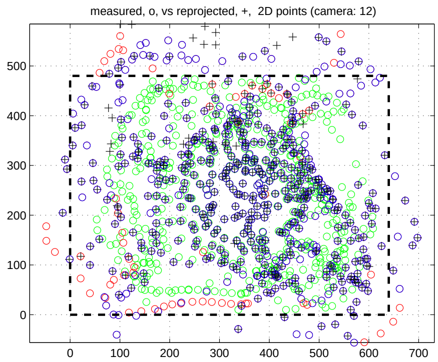


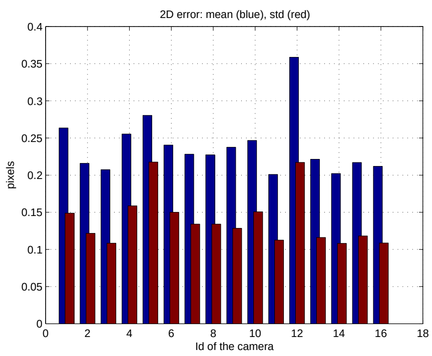
Results — Linear model





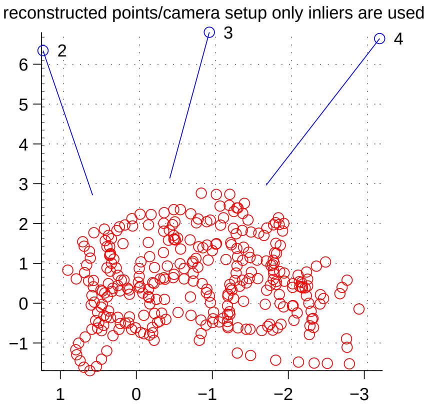
Results — Complete model

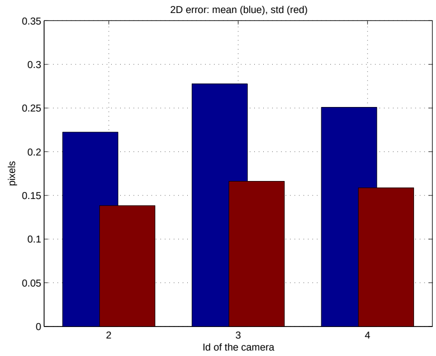




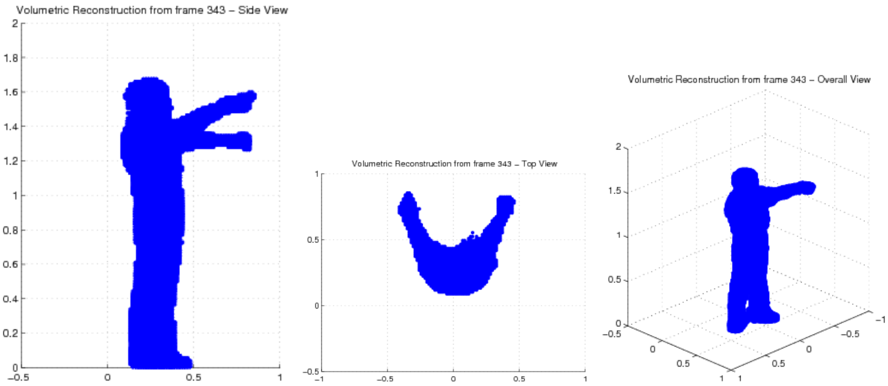
Very fine results from (almost) nothing!

Results — Simple setup





Application example — volumetric reconstruction



I know, it is just toy example. Still, it shows that the metric is OK.

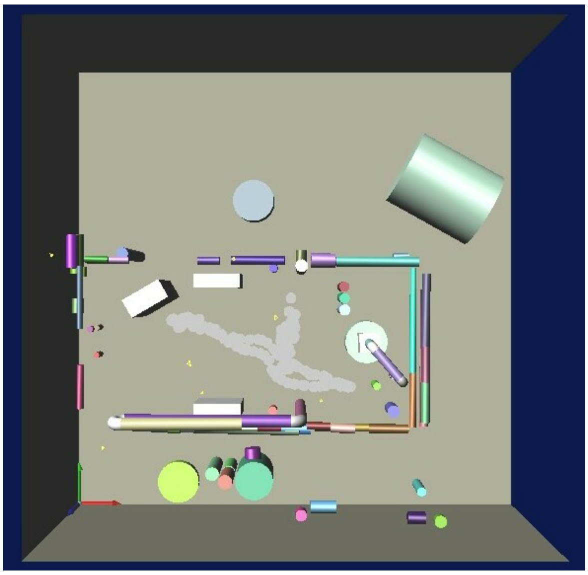
Application example — mobile multicamera setup

Video

Mobile multicamera setup - worker 3D tracking

Video

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