Robust Absolute and Relative Pose Estimation of a Central Camera System from 2D-3D Line Correspondences

Hichem Abdellali, Robert Frohlich, Zoltan Kato

Institute of Informatics, University of Szeged, Hungary

Research Group on Visual Computation

Problem Statement

- Estimate the **absolute** and **relative** pose of a camera system composed of **general central projection** cameras such as **perspective** and **omnidirectional** cameras
- Using **3D-2D line-pairs** (line features are an attractive alternative to point matches in urban scenarios due to the repetitive structures in road scenes)
- Useful for a wide range of computer vision applications like SLAM, 2D-3D fusion
- None of the existing methods estimate full absolute and relative poses in a multi-view system with omnidirectional and perspective cameras.

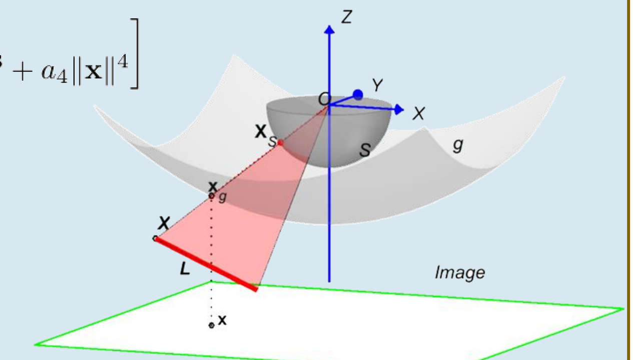
Camera Model

- [1] represents central cameras as a projection onto the surface of a unit sphere $\mathcal{S}$
- The camera coordinate system is in the center of $\mathcal{S}$ and the Z axis is the optical axis of the camera which intersects the image plane in the principal point

Knowing the internal calibration of the camera allows us to work directly with spherical image points $\mathbf{x}_\mathcal{S}$

The image plane maps to the surface of the sphere by:

1. lifting the image point onto the g surface: $\mathbf{x}_g = \begin{bmatrix} \mathbf{x} \\ a_0 + a_2\|\mathbf{x}\|^2 + a_3\|\mathbf{x}\|^3 + a_4\|\mathbf{x}\|^4 \end{bmatrix}$
2. centrally projecting onto the unit sphere S: $\mathbf{x}_\mathcal{S} = \dfrac{\mathbf{x}_g}{\|\mathbf{x}_g\|}$



A 3D world point X is projected into $\mathbf{x}_\mathcal{S} \in \mathcal{S}$ by a simple central projection taking into account the pose: $\mathbf{x}_\mathcal{S} = \dfrac{\mathbf{RX}+\mathbf{t}}{\|\mathbf{RX}+\mathbf{t}\|}$

Note:

the image point of a perspective camera can be represented on S by $\mathbf{x}_K = \mathbf{K}^{-1}\mathbf{x}$ **and** $\mathbf{x}_\mathcal{S} = \mathbf{x}_K/\|\mathbf{x}_K\|$

Projection of Lines and Camera Pose

- 3D line is represented as L = (**V**, **X**)
- **X**: a point on the line
- n = $[n_1, n_2, n_3]$: normal of the projection plane
- **V** = $[v_1, v_2, v_3]$: unit direction vector of the line

- We used **two** equations based on **two** different geometric observations

1. the **direction vector V** of the 3D line and **the unit normal n** of the projection plane is **perpendicular** ➔ $\mathbf{n}^\top\mathbf{R}\mathbf{V} = 0$ ➔ *only rotation*
2. the vector from the camera center ***C*** to an arbitrary point **X** on the 3D line L is also lying on the projection plane, thus it is also perpendicular to **the normal of the projection plane** ➔ $\mathbf{n}^\top(\mathbf{RX}+\mathbf{t}) = 0$ ➔ *rotation* ➔ *translation*

For *N* central cameras ➔

$$\mathbf{n}_i^\top \mathbf{R}_i \mathbf{R} \mathbf{V} = 0$$
$$\mathbf{n}_i^\top (\mathbf{R}_i(\mathbf{RX}+\mathbf{t}) + \mathbf{t}_i) = 0$$

Proposed Solution

Cayley Parametrization of 3D Rotations

- To get rid of the trigonometric functions, we use the Cayley transform to obtain a parametrization of the rotation matrix $\mathbf{R}$ in terms of 3 parameters b = $[b_1, b_2, b_3]^T$
- The correspondence $\mathbf{R} \leftrightarrow [\mathrm{b}]_\times$ is a one-to-one map between skew-symmetric matrices (represented as 3-vectors) and 3D rotations
- **Finding b involves having the rotation matrix** $\mathbf{R}$

$$(1+\mathbf{b}^\top\mathbf{b})\mathbf{R} = (1-\mathbf{b}^\top\mathbf{b})\mathbf{I} + 2[\mathbf{b}]_\times + 2\mathbf{b}\mathbf{b}^\top = \begin{bmatrix} 1+b_1^2-b_2^2-b_3^2 & 2b_1b_2-2b_3 & 2b_1b_3+2b_2 \\ 2b_1b_2+2b_3 & 1-b_1^2+b_2^2-b_3^2 & 2b_2b_3-2b_1 \\ 2b_1b_3-2b_2 & 2b_2b_3+2b_1 & 1-b_1^2-b_2^2+b_3^2 \end{bmatrix}$$

Line Back-projection Error on the Unit Sphere $\mathcal{S}$

- uses image line segments and corresponding 3D lines represented by its endpoint spherical coordinates $(\mathbf{a},\mathbf{b})$ and $(\mathbf{A},\mathbf{B})$ respectively
- $\delta(\mathbf{p})$ is the shortest orthodromic distance from a point p on $(\mathbf{a},\mathbf{b})$ to the back-projected 3D line $(\mathbf{A},\mathbf{B})$

\- n is the unit normal vector of the projection plane of $(\mathbf{A},\mathbf{B})$

\- λ is the great-circle length of $(\mathbf{a},\mathbf{b})$

$$\delta(\mathbf{p}) = \arctan\left(\frac{|\mathbf{n}\cdot\mathbf{p}|}{\|\mathbf{n}\times\mathbf{p}\|}\right) = \arctan\left(\frac{\|\mathbf{n}\|\|\mathbf{p}\||\sin\theta|}{\|\mathbf{n}\|\|\mathbf{p}\||\cos\theta|}\right) = \theta$$

The error function $\frac{1}{\lambda}(\delta^2(\mathbf{a}) + \delta^2(\mathbf{b}))$ characterizes the line projection error as the distance between $(\mathbf{a},\mathbf{b})$ and $(\mathbf{A},\mathbf{B})$

Normalization

$$s = \frac{1}{max(|h|,|w|,|d|)}.$$

- 2D image data is normalized by definition as we work on the unit sphere
- 3D line data is transformed into a unit cube, by a translation to the origin and uniform scaling factor s
- The result pose $(\tilde{\mathbf{R}}, \tilde{\mathbf{t}})$ needs to be denormalized

Minimal Solver

The minimal set consists of 3 line-pairs, providing 3 equations for the rotation only

$$\mathbf{c}^\top\mathbf{x} = \begin{bmatrix} n_1v_1+n_2v_2+n_3v_3 \\ 2n_1v_2+2n_2v_1 \\ 2n_1v_3+2n_3v_1 \\ 2n_2v_3+2n_3v_2 \\ n_1v_1-n_2v_2-n_3v_3 \\ -n_1v_1+n_2v_2-n_3v_3 \\ -n_1v_1-n_2v_2+n_3v_3 \\ -2n_2v_3+2n_3v_2 \\ 2n_1v_3-2n_3v_1 \\ -2n_1v_2+2n_2v_1 \end{bmatrix}^\top \begin{bmatrix} 1 \\ b_1b_2 \\ b_1b_3 \\ b_2b_3 \\ {b_1}^2 \\ {b_2}^2 \\ {b_3}^2 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \mathbf{0}$$

Direct Least Squares Solver

For n > 3 line-pairs

Take the sum of squares of the nonlinear system constructed from the minimal solver and then minimize $E(\mathbf{b})$

$$E(\mathbf{b}) = \sum_{i=1}^{n}(\mathbf{c}_i^\top\mathbf{x})^2$$

$$\nabla E(\mathbf{b}) = \begin{bmatrix} \frac{\partial E(\mathbf{b})}{\partial b_1} \\ \frac{\partial E(\mathbf{b})}{\partial b_2} \\ \frac{\partial E(\mathbf{b})}{\partial b_3} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n \mathbf{d}_{\mathbf{b}_1}{}_i^\top \mathbf{x}_{\mathbf{b}_1} \\ \sum_{i=1}^n \mathbf{d}_{\mathbf{b}_2}{}_i^\top \mathbf{x}_{\mathbf{b}_2} \\ \sum_{i=1}^n \mathbf{d}_{\mathbf{b}_3}{}_i^\top \mathbf{x}_{\mathbf{b}_3} \end{bmatrix} = \mathbf{0}$$

For each line pair d_{b1}, d_{b2} and d_{b3} can be expressed in terms of the coefficients (*n*,*V*)

Provides the Rotation $\mathbf{R}$

- The **translation** $\mathbf{t}$ is then obtained by back-substituting $\mathbf{R}$ into the linear system $\mathbf{n}^\top(\mathbf{RX}+\mathbf{t}) = 0$ which can be solved by SVD decomposition. The solver might have several solutions, it is choosing the geometrically valid one

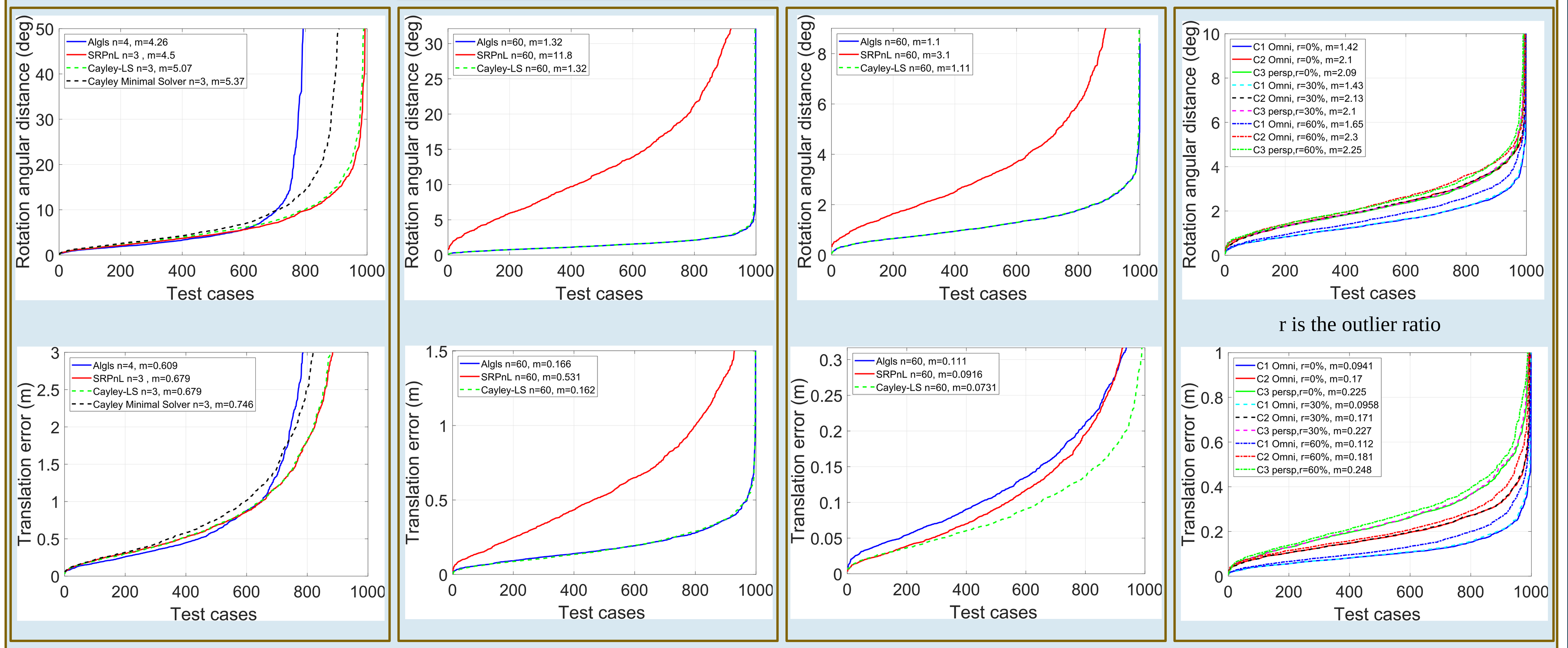
- **We used the automatic generator of Kukelova et al. [2] for a fair comparison in Matlab with competing methods, we also used Kneip's generator [3] to produce a solver in C++ (an order of magnitude faster)**

Evaluation on Synthetic Data

- Benchmark dataset of 1000 2D-3D synthetic images
- **3D scene**: 3 arbitrary planes with 20 lines on each
- **2D side**: we generated images of the scene by projecting the lines on perspective and omnidirectional cameras with 2378x1580 resolution and real camera parameters
- **2 + 1 cameras**: omni-perspective and perspective-omni configurations
- **Noise**: Corrupting one endpoint of the line (similarly in 2D and 3D), essentially adding a random number to each coordinate of the point
  - Perspective 7% and 15% 2D and 3D
  - Omnidirectional 7% and 10% 2D and 3D

Synthetic Data Plots

Minimal case **2D noise** (7%) | **Least-Squares solver n=60** **2D noise** (15%) | **Least-Squares solver n=60** **3D noise** (15%) | **RANSAC** with n=60, 2 omni 1 perspective configuration, **2D noise** (10%,15%)



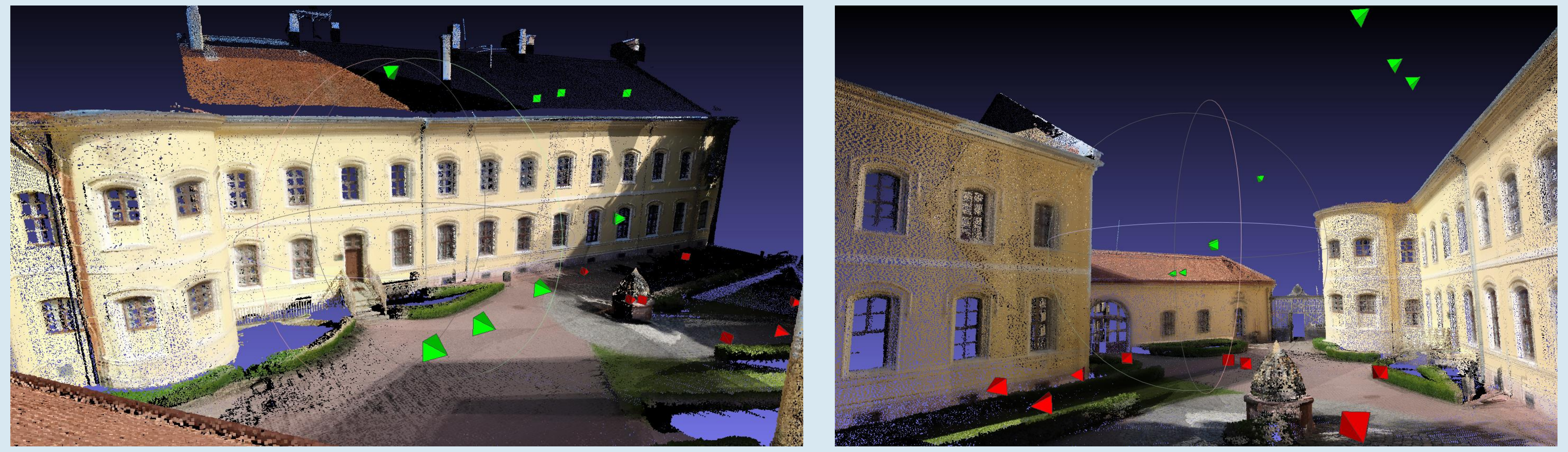
	Cayley Minimal Solver	Cayley-LS	ALgLs	SRPnL
Minimal case	0.0019	0.0104	0.0079	0.0031
60 lines	#	0.0093	0.0092	0.0022

Time in seconds, median over 1000 test cases with 2D noise (ALgLS requires at least 4 lines)

RANSAC: while a higher than 50% outlier ratio can be filtered out robustly, it drastically increases the execution time:

- with 30% outlier ratio is 0.14 (s)
- with 60% outlier ratio reaches 1.21 (s)

Real Data Results



Fusion result shown as colorized pointcloud with estimated Omni (red) and perspective (green) camera positions illustrated

References

[1] D. Scaramuzza et al. A Toolbox for Easily Calibrating Omnidirectional Cameras. **IROS -2006**

[2] Z. Kukelova et al. Automatic generator of minimal problem solvers. **ECCV-2008**. Springer.

[3] L. Kneip. Polyjam, 2015 [online]. https://github.com/laurentkneip/polyjam.

Paper

Acknowledgements

This work was partially supported by the NKFI-6 fund through project K120366; "Integrated program for training new generation of scientists in the fields of computer science", EFOP-3.6.3-VEKOP-16-2017-0002; the Research & Development Operational Programme for the project "Modernization and Improvement of Technical Infrastructure for Research and Development of J. Selye University in the Fields of Nanotechnology and Intelligent Space", ITMS 26210120042, co-funded by the European Regional Development Fund; Research & Innovation Operational Programme for the Project: "Support of research and development activities of J. Selye University in the field of Digital Slovakia and creative industry", ITMS code: NFP313010T504, co-funded by the European Regional Development Fund.

Research Group

ICCV Workshop on Computer Vision for Road Scene Understanding and Autonomous Driving (CVRSUAD 2019), October 27 – November 02, 2019, Seoul, Korea