Computing Rolling Shutter Camera Pose via Optimized Algebraic Geometry

Tomas Pajdla

C. Albl, Z. Kukelova, V. Larsson, A. Sugimoto





CTU CZECH TECHNICAL UNIVERSITY IN PRAGUE

Czech Technical University in Prague

ETH Zürich

NII 大学共同利用機関法人 情報・システム研究機構 国立情報学研究所 National Institute of Informatics

What is the Rolling Shutter Effect?

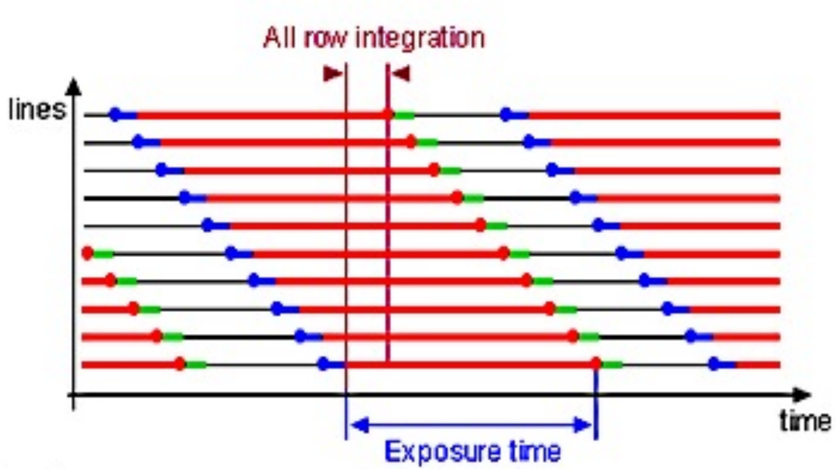
GS - Global shutter

RS - Rolling shutter (most cameras)

youtu.be/7TGKFdrY9aw

How does the Rolling Shutter work?

- Images scanned line by line



- The effect

The good

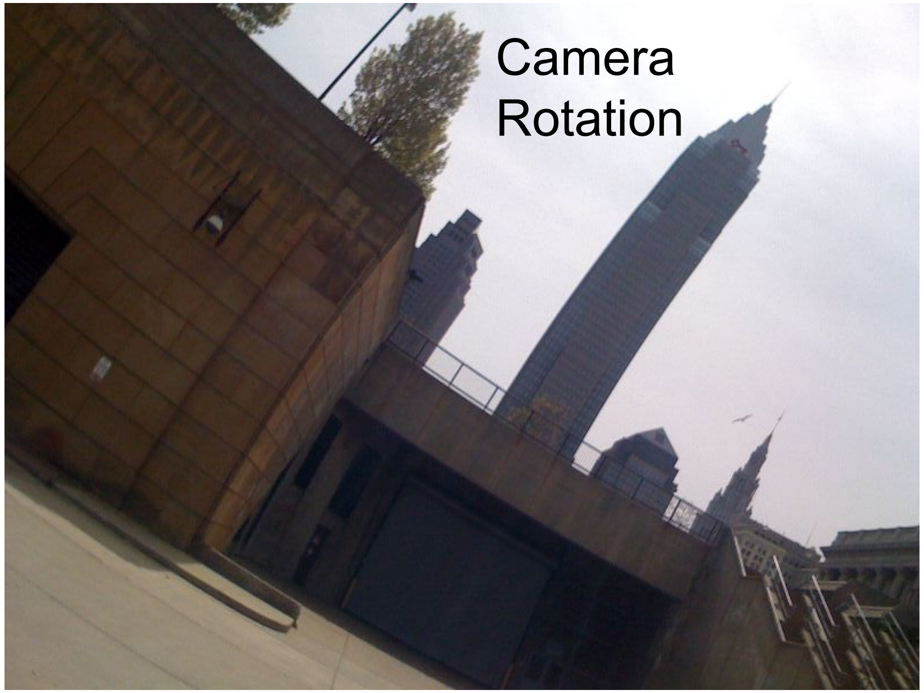
- Higher frame rate
- Longer exposure time
- Cheaper and easier to manufacture

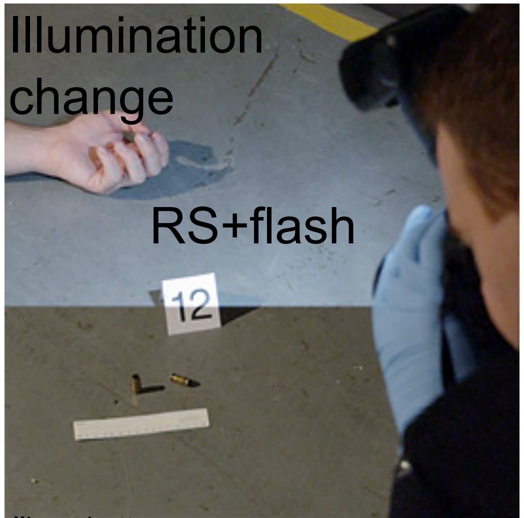
The bad

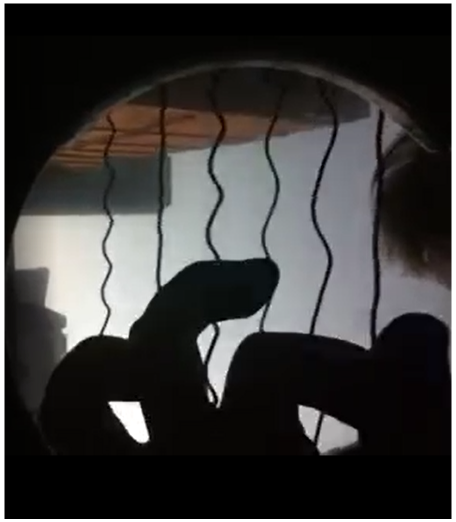
- Image distortions
- Non-perspective projections

How does the Rolling Shutter look?

And the ugly…







String
oscillations

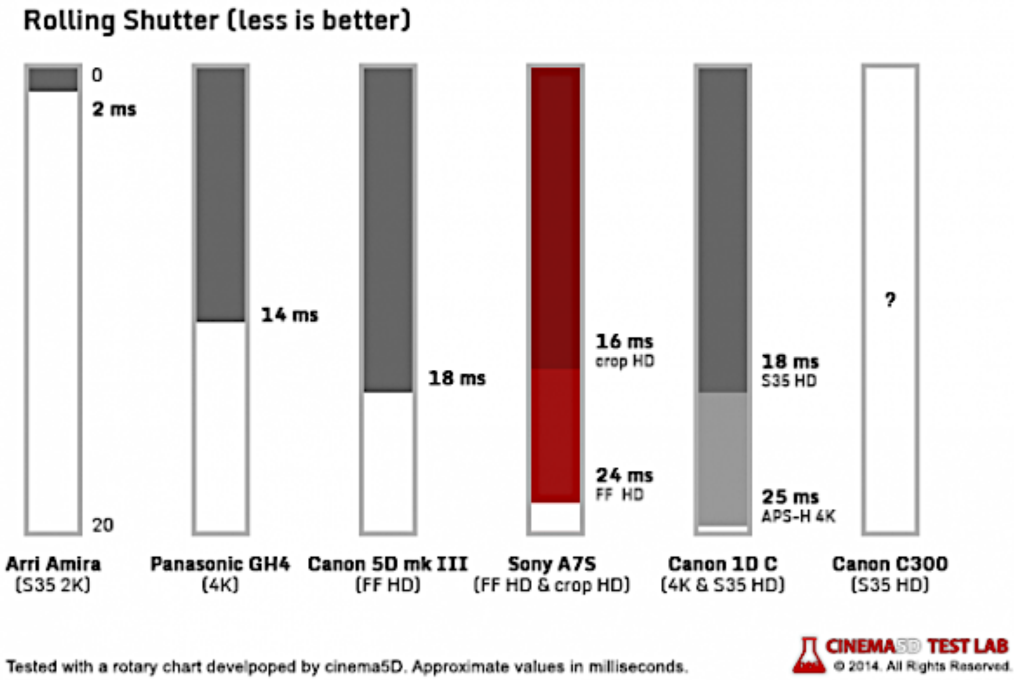
http://www.red.com/learn/red-101/global-rolling-shutter

Rolling shutter is everywhere

- Most of cameras: cellphones, industrial cams, … professional DSLR



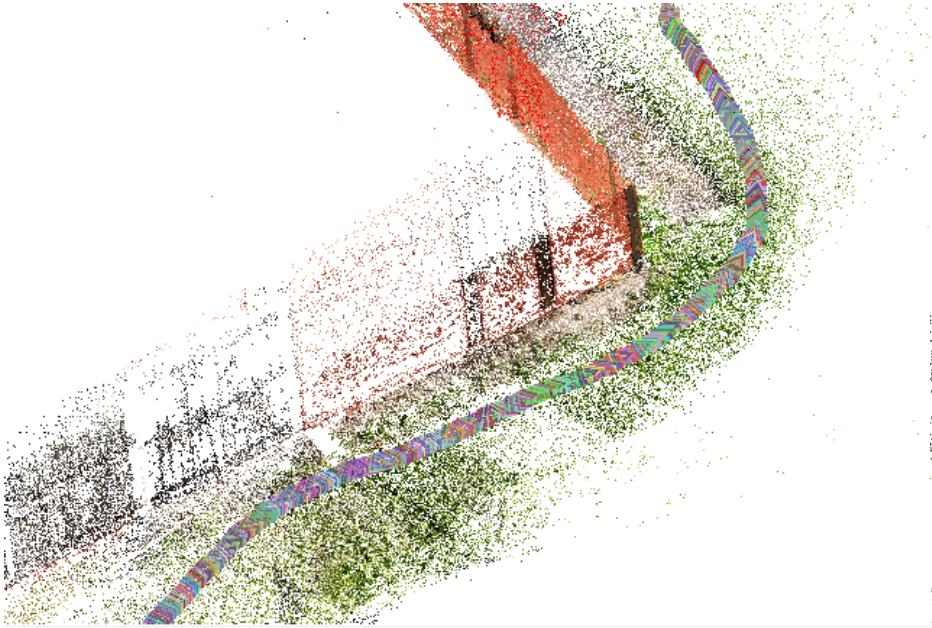

- Affects both videos AND single images
  - Difference between top and bottom can be ~1/30s



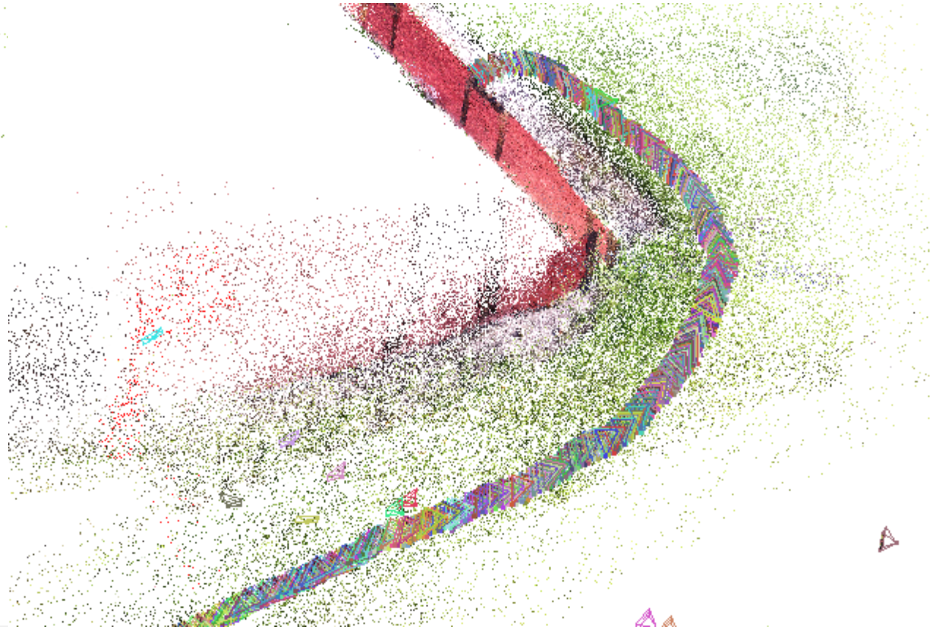
3D Reconstruction with Rolling Shutter

3D reconstruction from RS images ... degraded if ignored

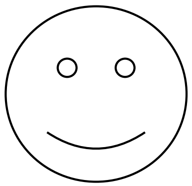
Feature detection ⇨ Matching ⇨ Relative pose ⇨ Absolute pose ⇨ Triangulation ⇨ Bundle adjustment

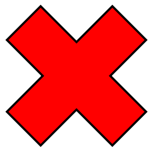


Global shutter (Canon)

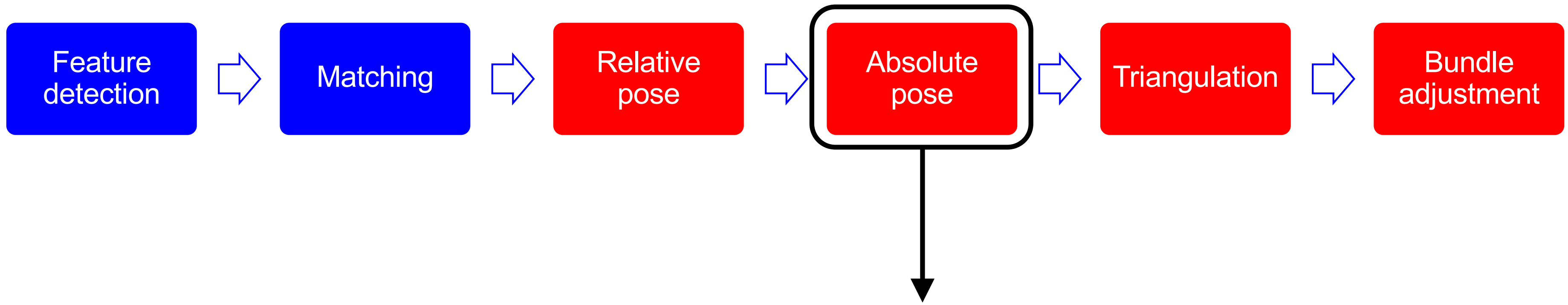


Rolling shutter (iPhone 4)





Absolute Camera Pose with Rolling Shutter

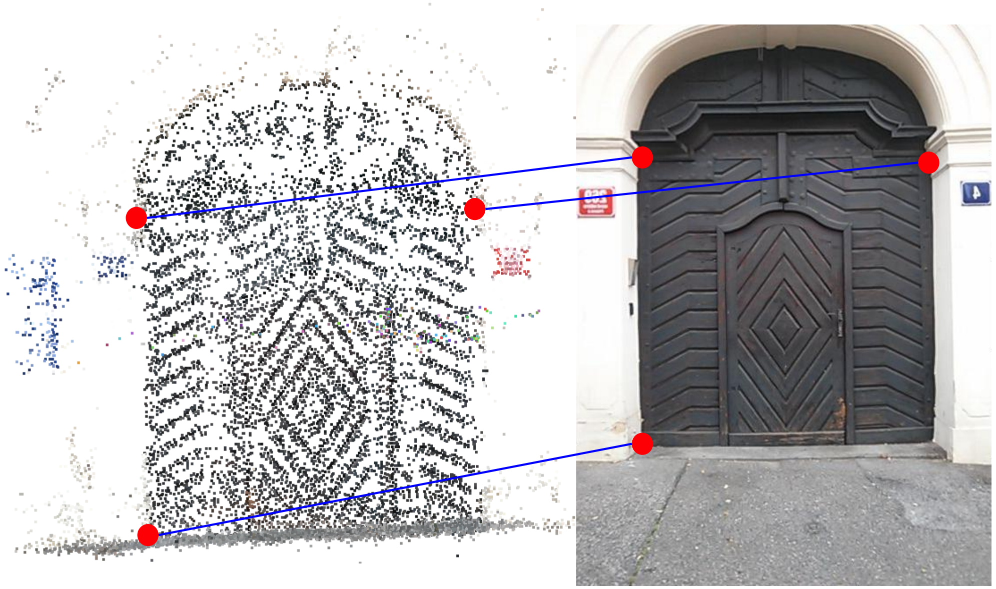


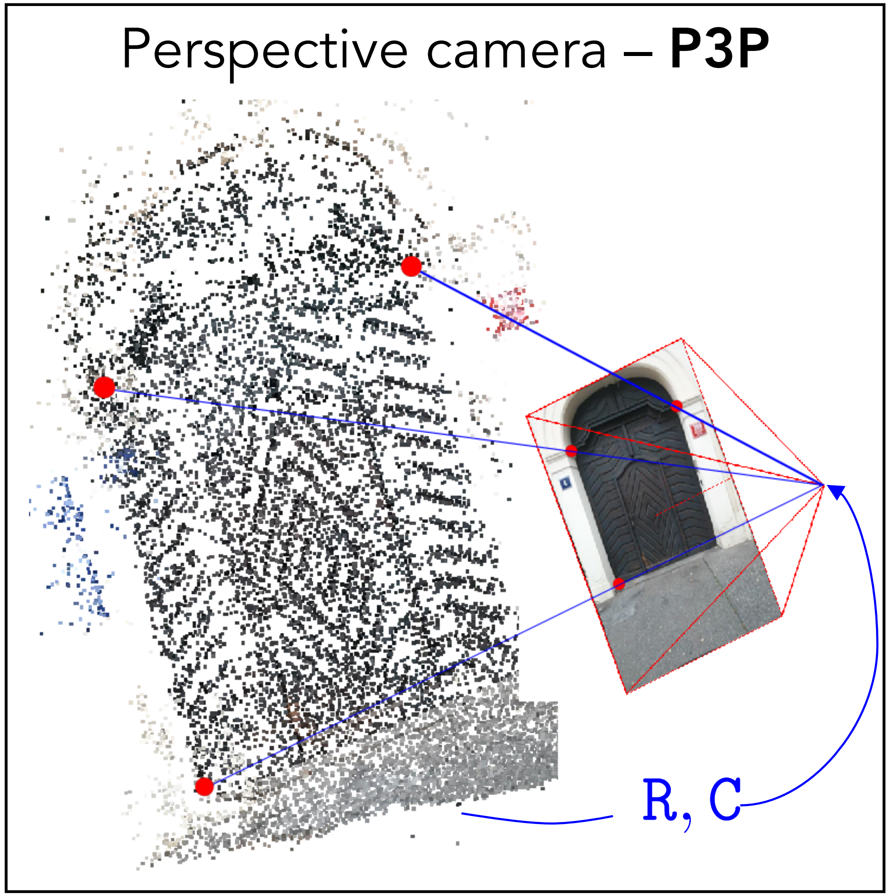
Absolute camera pose with RS

1. *C. Albl, Z. Kukelova, T Pajdla.*
   R6P - Rolling Shutter Absolute Camera Pose*. CVPR 2015*
2. *C. Albl, Z. Kukelova, T Pajdla.*
   RS Absolute Camera Pose Problem with known Vertical Direction. *ICCV 2015*
3. *Z Kukelova, C Albl, A Sugimoto, T Pajdla.*
   Linear solution to the minimal absolute pose rolling shutter problem*. ACCV 2018*
4. *C Albl, Z Kukelova, V Larsson, T Pajdla.*
   Rolling Shutter Camera Absolute Pose. TPAMI 2019

Absolute Camera Pose with Rolling Shutter

3D points ⟺ 2D points



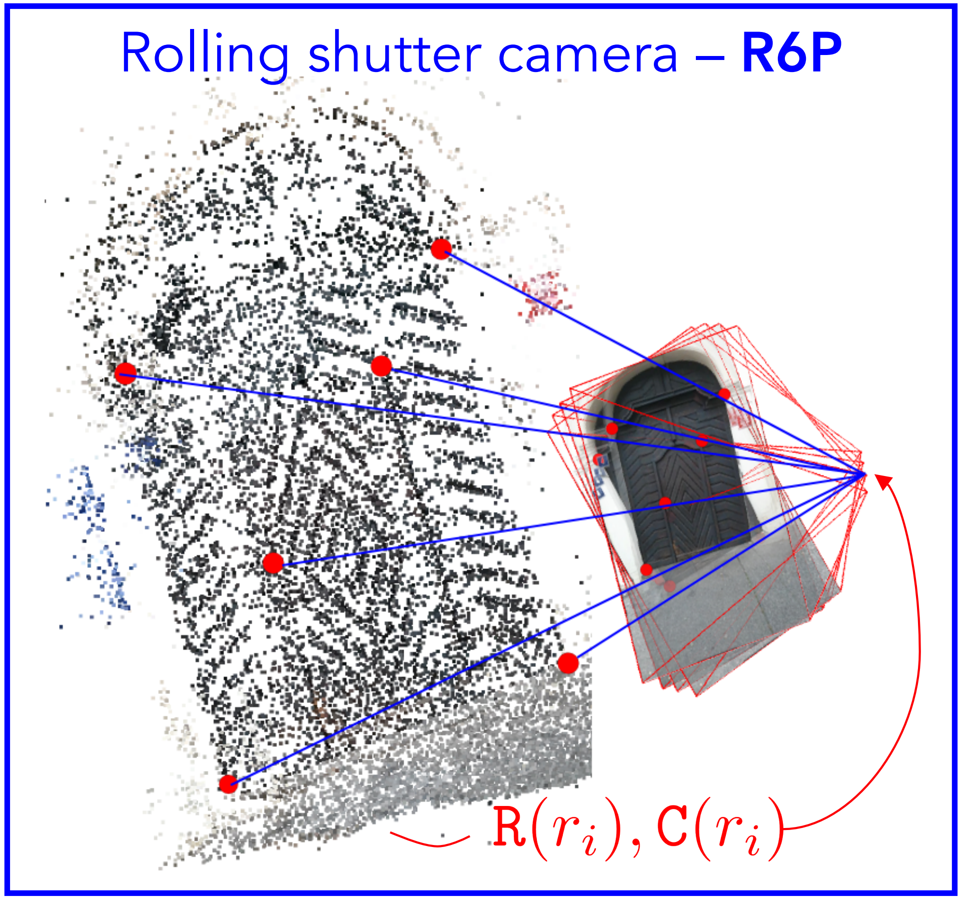


3 correspondences

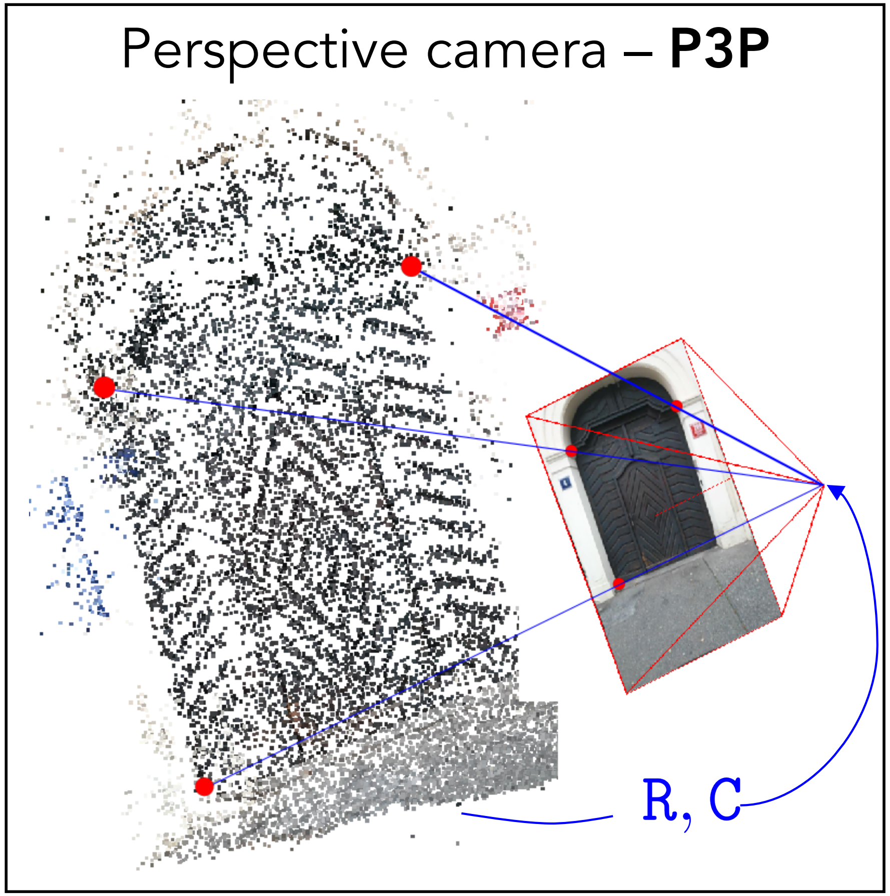
[Haralick CVPR 1991] [Quan PAMI 1999]
[Triggs IJCV 1999] [Wut JMIV 2006]
[Zhi MMRC 2002] [Lepetit IJCV 2009]

Absolute Camera Pose with Rolling Shutter

This work = R6P



6 correspondences

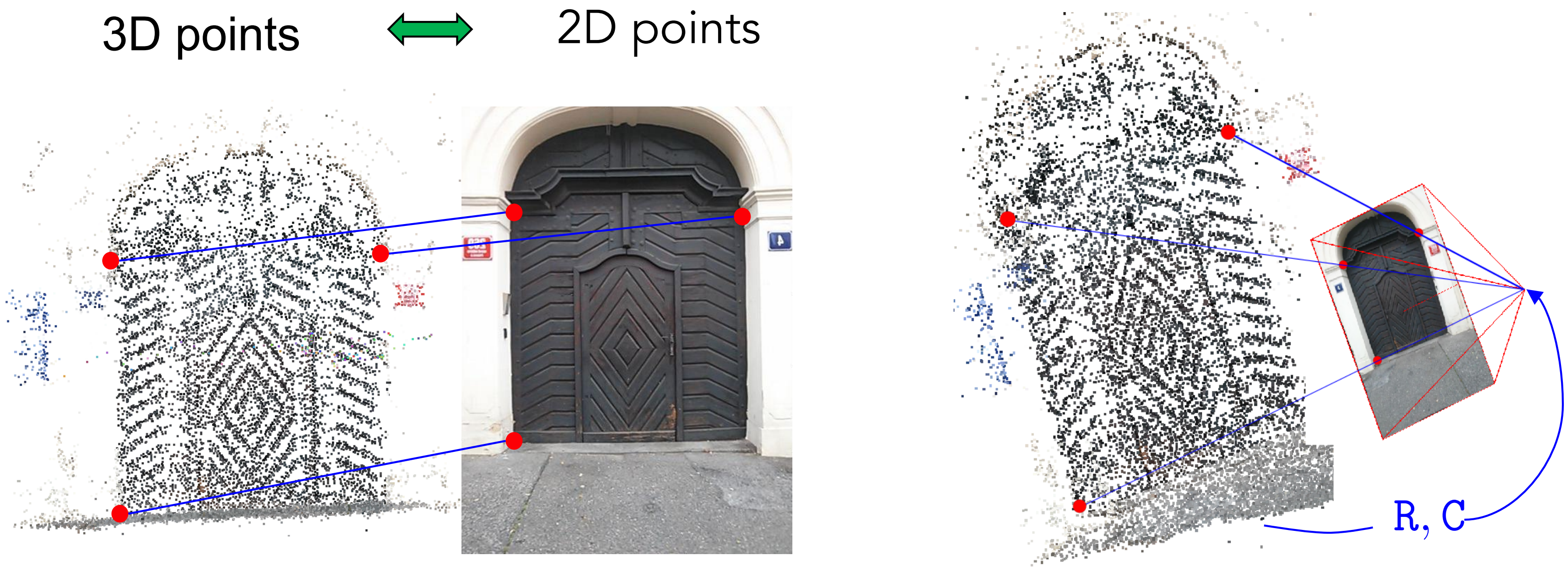


3 correspondences

[Haralick CVPR 1991][Quan PAMI 1999]
[Triggs IJCV 1999][Wut JMIV 2006]
[Zhi MMRC 2002][Lepetit IJCV 2009]

RANSAC: Optimization scheme to deal with gross errors

Enumerating all subsets replaced by checking only some of them



repeat

1. Generate random tuples of 2D-3D matches
2. Compute R, C by solving algebraic equations
3. Count the number of good matches

many trials
be fast

Return the largest set of good matches

Why to be fast?

- Many samples needed to be sure to find a good sample!

To find a gross-error-free sample with 95% probability we have to try at least the following number of samples:

Gross-error-free data fraction [%]

Sample size

	15%	20%	30%	40%	50%	70%
2	132	73	32	17	10	4
4	5916	1871	368	116	46	11
7	$1.75 \cdot 10^6$	$2.34 \cdot 10^5$	$1.37 \cdot 10^4$	1827	$\boxed{382}$	35
8	$1.17 \cdot 10^7$	$1.17 \cdot 10^6$	$4.57 \cdot 10^4$	4570	765	50
12	$2.31 \cdot 10^{10}$	$7.31 \cdot 10^8$	$5.64 \cdot 10^6$	$1.79 \cdot 10^5$	$1.23 \cdot 10^4$	215
18	$2.08 \cdot 10^{15}$	$1.14 \cdot 10^{13}$	$7.73 \cdot 10^9$	$4.36 \cdot 10^7$	$7.85 \cdot 10^5$	1838
30	∞	∞	$1.35 \cdot 10^{16}$	$2.60 \cdot 10^{12}$	$3.22 \cdot 10^9$	$1.33 \cdot 10^5$
40	∞	∞	∞	$2.70 \cdot 10^{16}$	$3.29 \cdot 10^{12}$	$4.71 \cdot 10^6$

Solving time: **micro-mili** seconds

How to be fast?

How to be fast?

1. Specialized solving methods

2. Assume generic data

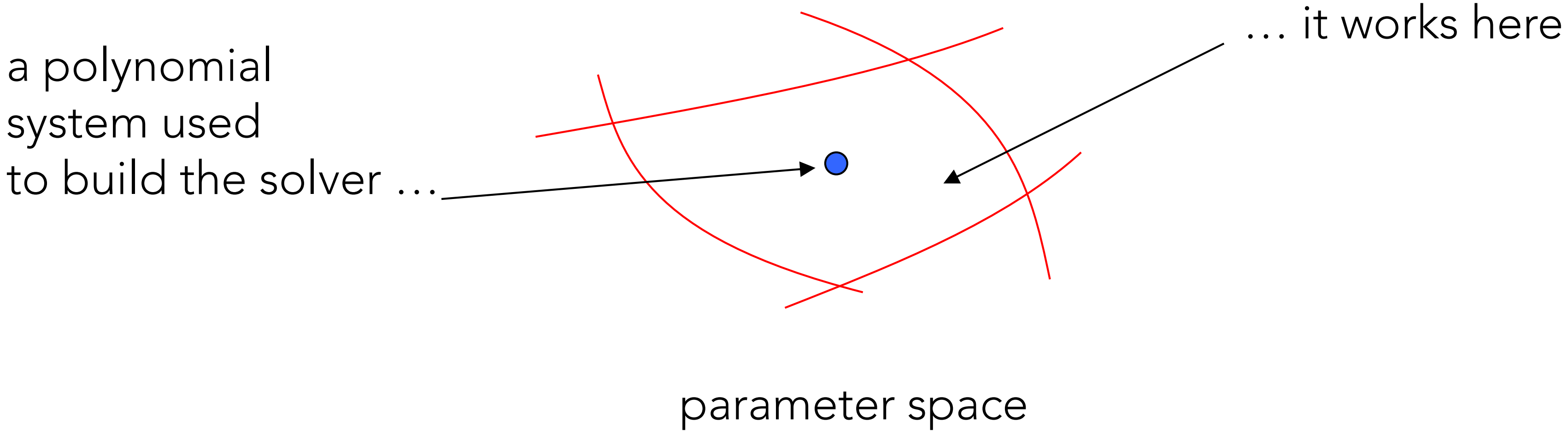
3. Use tricks, optimize, hard code, …

Many problems are generic

Solvers do not (much) differ from one problem to another.

$\rightarrow$ Solver is made out by solving a single concrete system and then used on other systems

$\rightarrow$ This works around generic solutions

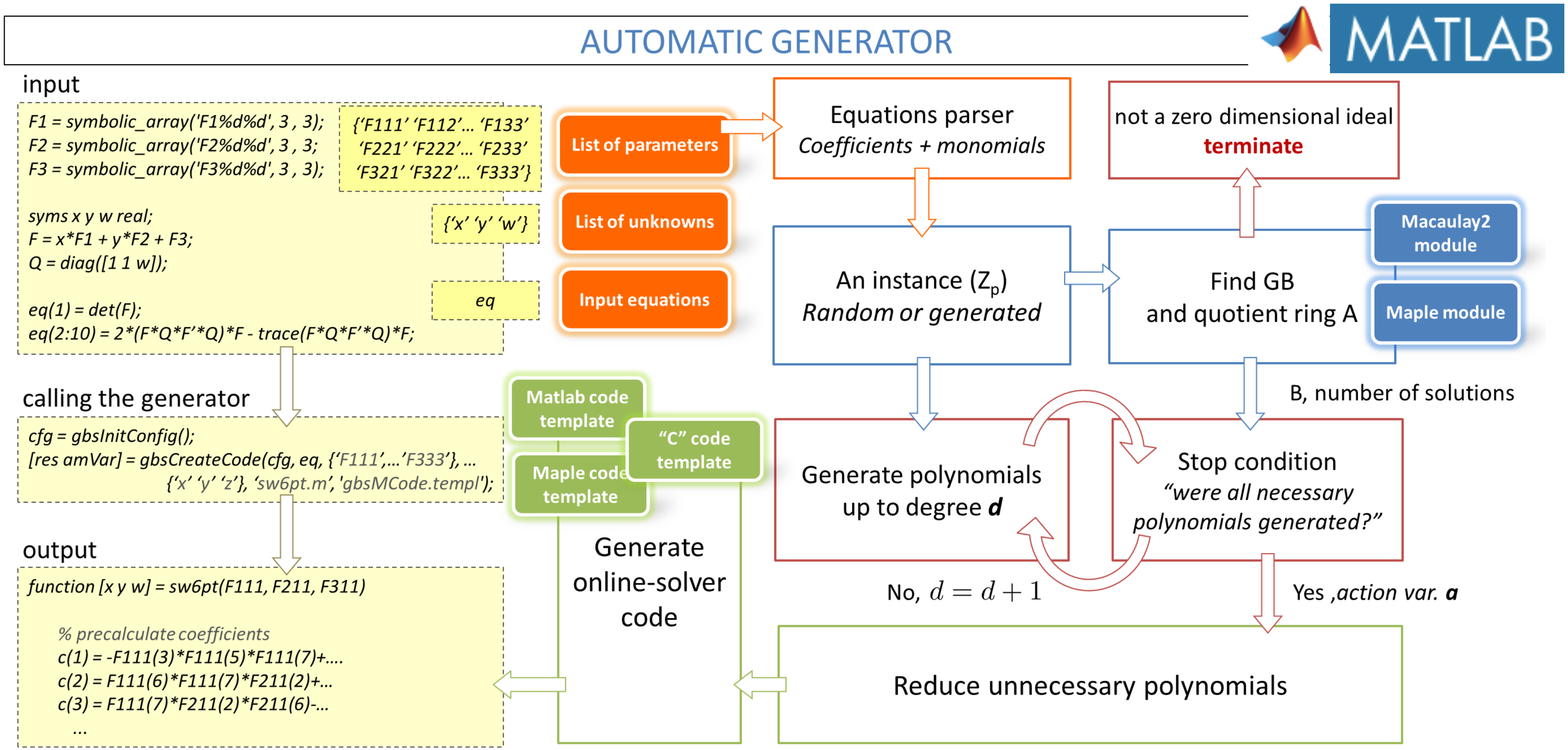


Strategy of fast solving

Offline phase (may be slow)

1. Fabricate a concrete generic example of a polynomial system (generating 0-dim radial ideal I)
2. Analyze the system by a generic method (Macaulay2, FGb, …) to get the degree, (standard monomial) basis in R/I, …
3. Create an elimination template for constructing a multiplication matrix $\mathtt{M}_f$ of multiplication by a suitable polynomial $f \in \mathbb{C}[x_1, \ldots, x_n]$ (an unknown) in a finite-dimensional factor ring $A = \mathbb{C}[x_1, \ldots, x_n]/I$.
4. Implement efficiently in floating points, optimize, test, … (vary ordering, basis selection, …)

Automatic generator of “minimal solvers”



- Z Kukelova, M Bujnak, T Pajdla.
  Automatic Generator of Minimal Problem Solvers. ECCV 2008.
- V Larsson, K Astrom, M Oskarsson.
  Efficient Solvers for Minimal Problems by Syzygy-Based Reduction. CVPR 2017.
- V Larsson, M Oskarsson, K Astrom, A Wallis, Z Kukelova, T Pajdla.
  Beyond Grobner Bases: Basis Selection for Minimal Solvers. CVPR 2018

Strategy of fast solving

Online (must be fast)

1. Fill the elimination template to get matrix $\mathtt{M}_f$
2. Solve numerically by finding eigenvectors of $\mathtt{M}_f$
   (or get a univariate poly and use real root bracketing)

Rolling Shutter Camera Projection

Standard (calibrated) perspective projection

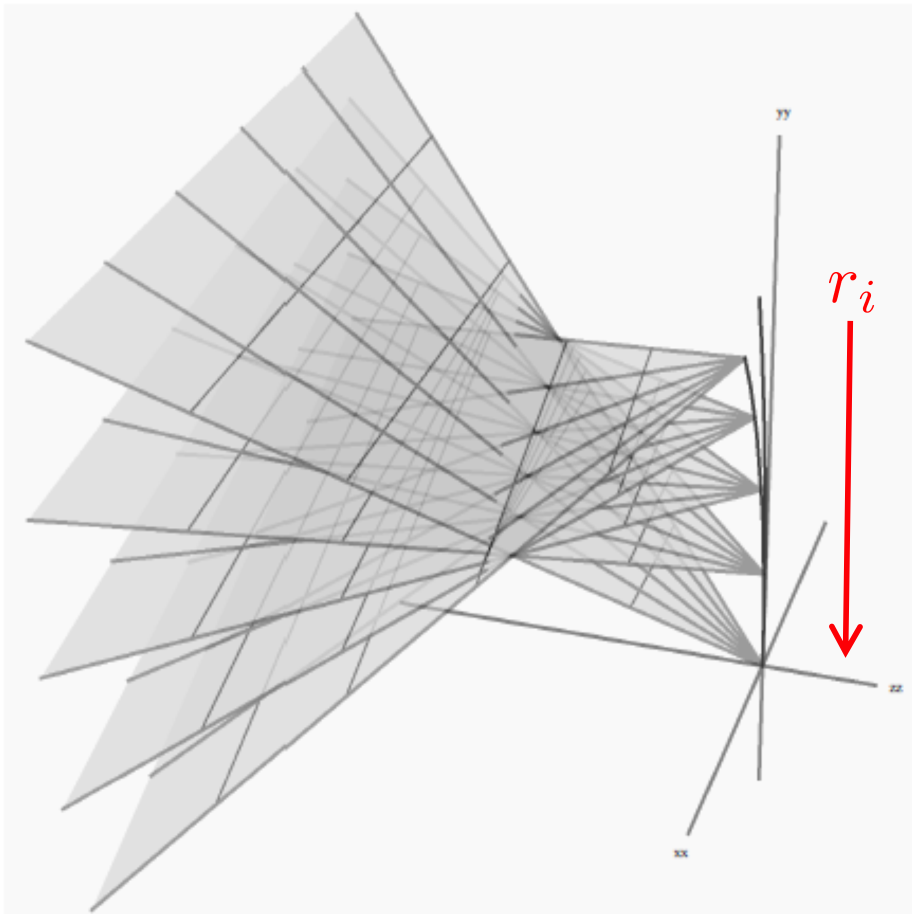
$$\lambda_i \mathtt{x}_i = \mathtt{R}\mathtt{X}_i + \mathtt{C}$$

RS camera undergoing motion during im

$$\lambda_i \mathtt{x}_i = \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = \mathtt{R}(r_i)\mathtt{X}_i + \mathtt{C}(r_i)$$

Camera pose changes for every row

How to model $\mathtt{R}(r_i)$ and $\mathtt{C}(r_i)$?



Picture from Meingast et al.

Rolling Shutter Camera Projection

$$\lambda_i \mathtt{x}_i = \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = \mathtt{R}(r_i)\mathtt{X}_i + \mathtt{C}(r_i)$$

Camera initial pose

$$\lambda_i \mathtt{x}_i = \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = \mathtt{R}_m(r_i)\mathtt{R}_0\mathtt{X}_i + \mathtt{C} + \mathtt{C}_m(r_i)$$

Motion during capture

Solving in general leads to complicated polynomials

We analyzed several models

- SLERP
- Cayley parameterization
- Linearized
- …
- **Double linear model**

$$\mathtt{C}_m(r_i) = (r_i - r_0)\mathtt{t}$$

[Hedborg CVPR-2012]

Rolling Shutter Double-Linearized Projection

Full projection model

$$\lambda_i \mathtt{x}_i = \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = \mathtt{R}_m(r_i)\mathtt{R}_0\mathtt{X}_i + \mathtt{C} + \mathtt{C}_m(r_i)$$

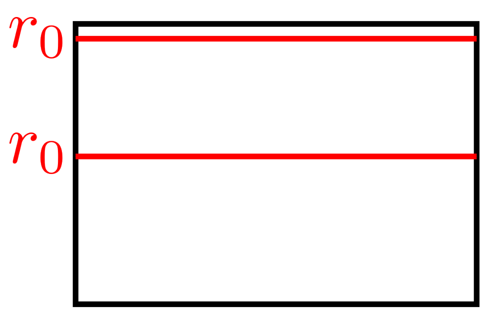
Double-linearized projection model

Camera initial pose

$$\lambda_i \boxed{\begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix}} = (\mathtt{I} + (r_i - r_0)[\mathtt{w}]_x)(\mathtt{I} + [\mathtt{v}]_x)\boxed{\mathtt{X}_i} + \mathtt{C} + (r_i - r_0)\mathtt{t}$$

Motion during capture

known



Constructing R6P Solver

Six 3D-2D correspondences

$$\lambda_1 \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = (\mathtt{I} + (r_i - r_0)[\mathtt{w}]_x)(\mathtt{I} + [\mathtt{v}]_x)\,\mathtt{X}_i + \mathtt{C} + (r_i - r_0)\mathtt{t}$$

$$\vdots$$

unknown

$$\lambda_6 \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = (\mathtt{I} + (r_i - r_0)[\mathtt{w}]_x)(\mathtt{I} + [\mathtt{v}]_x)\,\mathtt{X}_i + \mathtt{C} + (r_i - r_0)\mathtt{t}$$

18 equations

6 unknown scale parameters lambda

Constructing R6P Solver

Multiply by $\quad \mathtt{S} = \begin{bmatrix} 0 & -1 & c_i \\ 1 & 0 & -r_i \\ -c_i & r_i & 0 \end{bmatrix}$ to eliminate lambdas

$$\boxed{\mathtt{0} = \mathtt{S}((\mathtt{I} + (r_i - r_0)[\mathtt{w}]_x)(\mathtt{I} + [\mathtt{v}]_x)\mathtt{X}_i + \mathtt{C} + (r_i - r_0)\mathtt{t})}$$

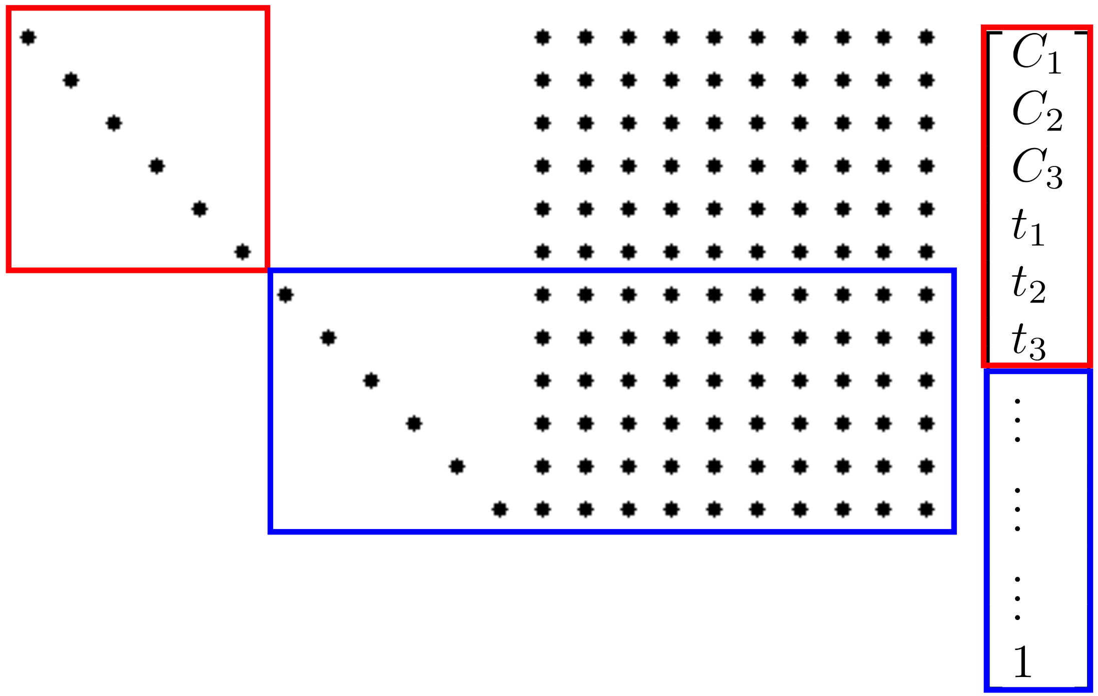
12 linearly independent equations (12x16 matrix … 16 monomials)

Matrix form

$$\begin{matrix}
\bullet & \bullet & \bullet & & \bullet & \bullet & \bullet & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & & \bullet & \bullet & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
& \bullet & \bullet & & \bullet & \bullet & \bullet & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & & \bullet & \bullet & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
& \bullet & \bullet & & \bullet & \bullet & \bullet & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
& \bullet & \bullet & & \bullet & \bullet & \bullet & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & & \bullet & \bullet & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
& \bullet & \bullet & & \bullet & \bullet & \bullet & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & & \bullet & \bullet & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
& \bullet & \bullet & & \bullet & \bullet & \bullet & & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & & \bullet & \bullet & & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet
\end{matrix}
\begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ t_1 \\ t_2 \\ t_3 \\ \vdots \\ \vdots \\ \vdots \\ 1 \end{bmatrix} = 0$$

Constructing R6P Solver

Simplify by Gauss-Jordan elimination



C and t linear

6 equations, 6 unknowns v & w (16 monomials)

Solve for v & w → back-substitution → C & t

Constructing R6P Solver

The remaining 16 monomials are bilinear in v and w

$$v_1, v_2, v_3, w_1, w_2, w_3, v_1w_1, v_1w_2, v_2w_1, v_1w_3, v_2w_2, v_3w_1, v_2w_3, v_3w_2, v_3w_3$$

We can write $\mathtt{M}(v)\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ 1 \end{bmatrix} = 0$, where $\mathtt{M}(v)$ is a 6x4 matrix

4x4 subdeterminants of $\mathtt{M}(v)$ must be zero

$\downarrow$

15 equations in 3 variables and 35 monomials

Use automatic generator of Gröbner basis solvers [Kukelova ECCV 2008] to solve for v

0.3ms in C++ (Eigen)

Double linearization … an initialization needed

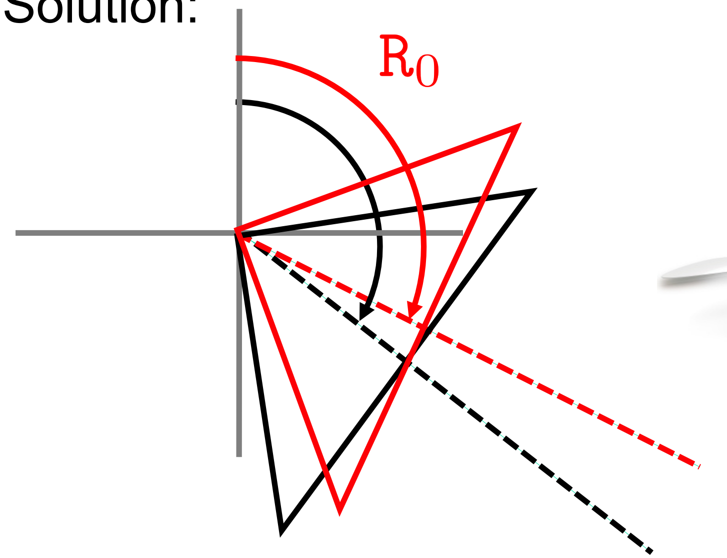
Linearization of rotation

$$\lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = (\mathtt{I} + (r_i - r_0)[\mathtt{w}]_x)(\mathtt{I} + [\mathtt{v}]_x)\mathtt{X}_i + \mathtt{C} + (r_i - r_0)\mathtt{t}$$

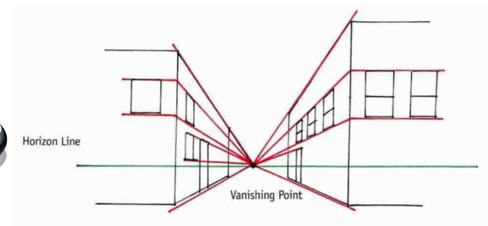
OK – small rotation during the capture

NOT OK – rotation can be arbitrary

Solution:



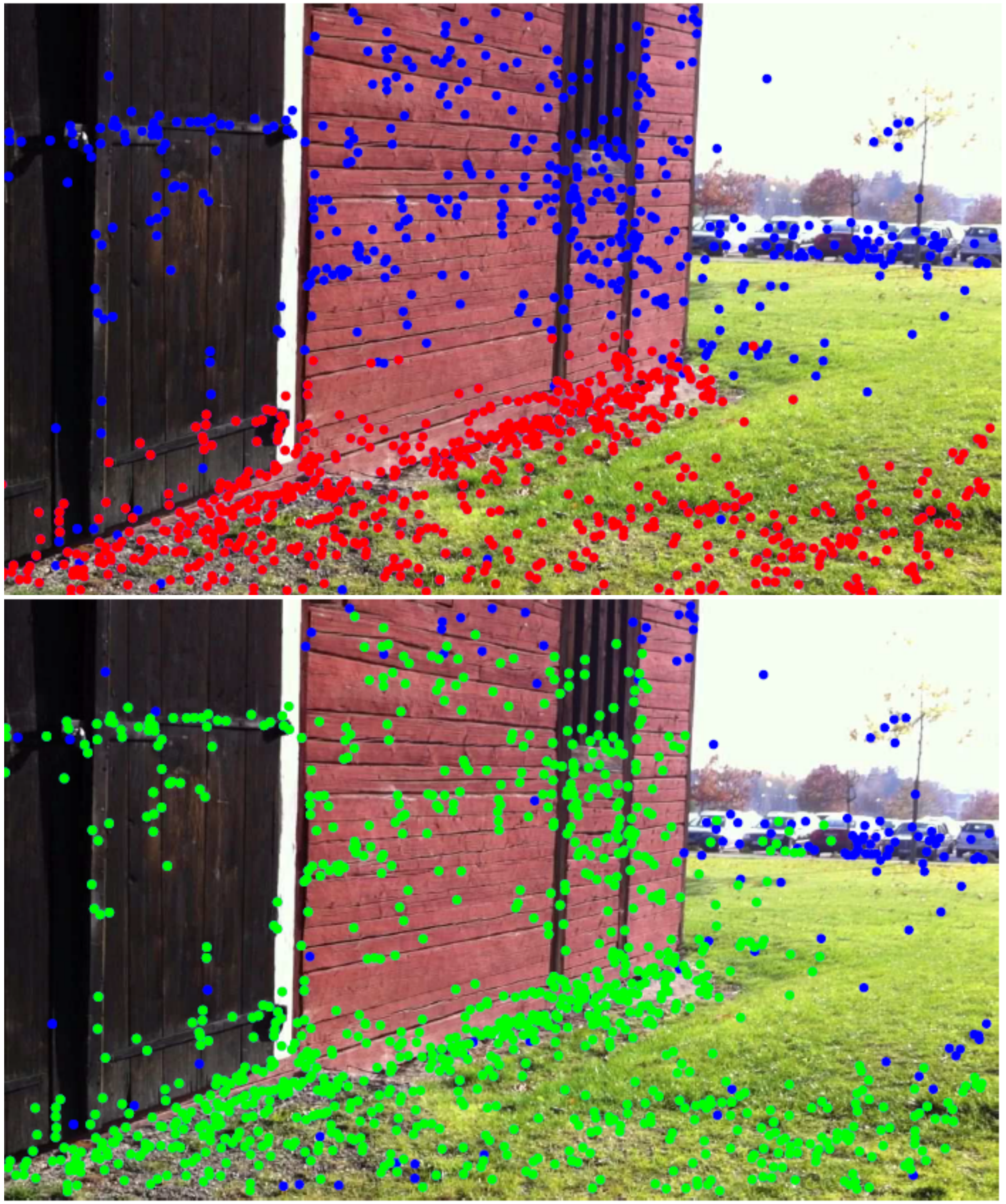
P3P → R6P



IMU → R6P

Real Experiments

P3P inliers
788

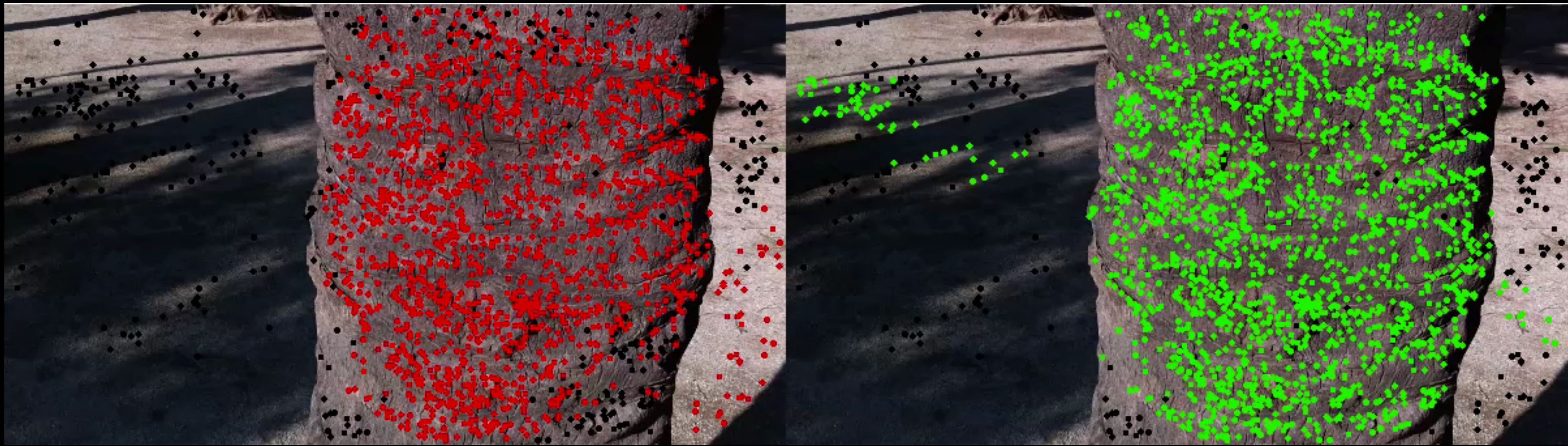


R6P inliers
1152

Data from
Hedborg et.al,
CVPR12

Real experiments

P3P (inliers in red) R6P (inliers in green)



Real Experiments

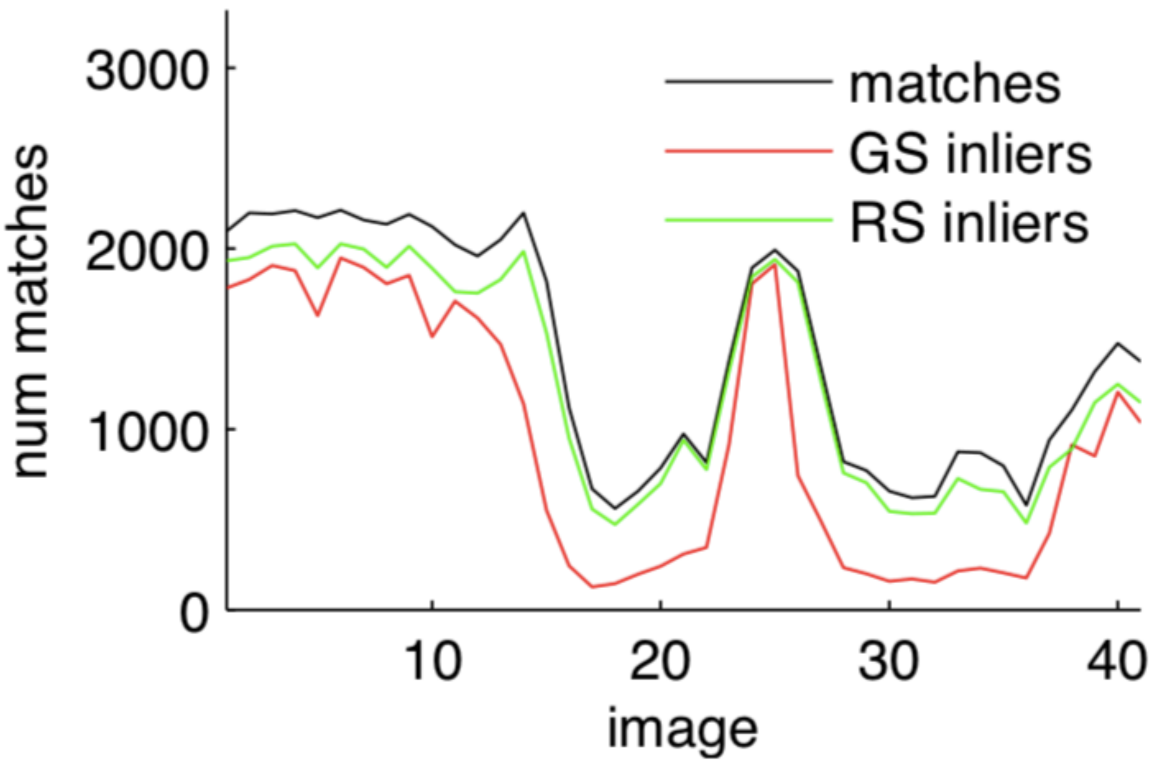


Fig. 5: Examples of experiments on real data. Number of inliers after running 1000 rounds of RANSAC, averaged over 100 RANSAC runs. Number of 2D-3D matches from global shutter images to rolling shutter images are in black, number of inliers obtained by P3P are in red and number of inliers obtained by R6P-2lin are in green. The results are averaged over 100 runs to reduce randomness.

Direct R6P without initialization

Cayley Parameterization

$$\lambda_i \boxed{\begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix}} = (\mathtt{I} + (r_i - r_0)[\mathtt{w}]_x)\mathtt{R}(\mathtt{v})\mathtt{X}_i + \boxed{\mathtt{C}} + (r_i - r_0)\mathtt{t} - r_0)\mathtt{t}$$

known

Motion during capture

$$\mathtt{R}(\mathtt{v}) = \frac{1}{1+v_1^2+v_2^2+v_3^2} \begin{bmatrix} 1+v_1^2-v_2^2-v_3^2 & 2\,v_1\,v_2 - 2\,v_3 & 2\,v_2 + 2\,v_1\,v_3 \\ 2\,v_3 + 2\,v_1\,v_2 & 1 - v_1^2 + v_2^2 - v_3^2 & 2\,v_2\,v_3 - 2\,v_1 \\ 2\,v_1\,v_3 - 2\,v_2 & 2\,v_1 + 2\,v_2\,v_3 & 1 - v_1^2 - v_2^2 + v_3^2 \end{bmatrix}$$

Corresponds to quaternion $w + iv_1 + jv_2 + kv_3$ where $w = 1$

v = rotation axis as a unit vector scaled by $\tan \alpha/2$ ➡ rotations by 180° prohibited

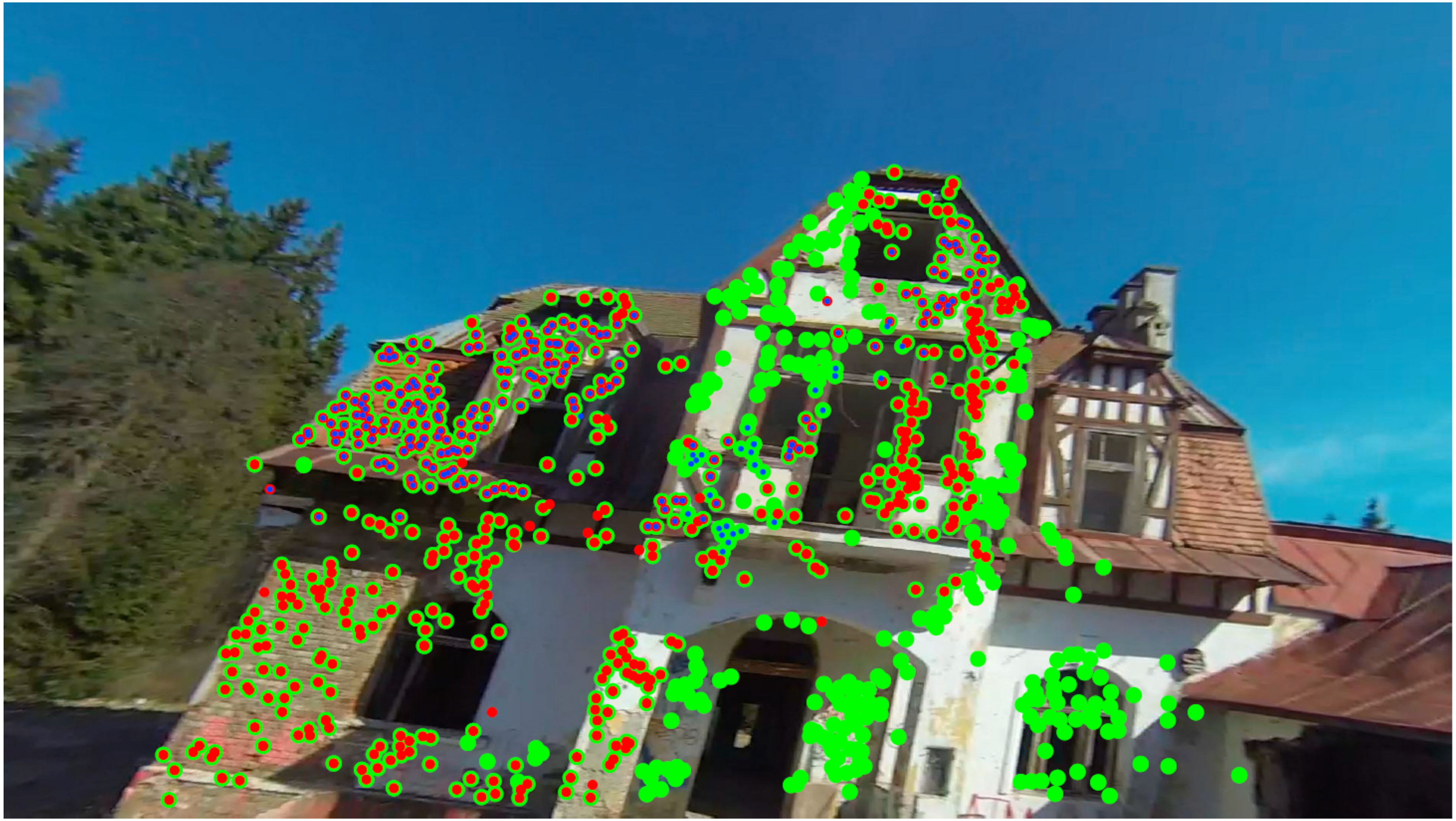
Direct R6P without initialization

- Eliminate C and t
- Write remaining equations in w and v as $\mathtt{M}(v) \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ 1 \end{bmatrix} = 0$
- Again the 15 determinants of $\mathtt{M}(v)$ must be 0
- Now 15 equations of degree 8 and 165 monomials
- PROBLEM – a family two-dimensional solutions introduced
- They correspond to $\mathtt{M}(v) \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ 0 \end{bmatrix} = 0$ and satisfy $1 + v_1^2 + v_2^2 + v_3^2$
- The new solutions are all complex and do not correspond to a valid R
- LUCKILY – all 15 equations divisible by $1 + v_1^2 + v_2^2 + v_3^2$
- Using [Larsson et.al. 2017] $\Rightarrow$ elimination template 99x163 and 64 solutions
- Using Sturm sequences to find the roots – 1.4 ms runtime for the whole solver

R6P vs non-linear RS refinement

- QUESTION – Can we just initialize with P3P and solve for RS parameters using non-linear optimization techniques?
- ANSWER – It depends

P3P P3P+LO R6P



R6P vs non-linear RS refinement

- QUESTION – Can we just initialize with P3P and solve for RS parameters using non-linear optimization techniques?
- ANSWER – It depends

Fraction of inliers (the higher, the better)

P3P+LO

R6P

	P3P		P3P LO-P		P3P LO-RS		P3P R6P-2lin		R6P-1lin		P3P R6P-2lin LO-RS		P3P LO-RS R6P-2lin LO-RS		R6P-1lin LO-RS	
	min	avg	min	avg	min	avg	min	avg	min	avg	min	avg	min	avg	min	avg
seq01	0.53	0.77	0.48	0.76	0.70	0.91	0.72	0.92	0.72	0.91	**0.77**	**0.94**	0.76	**0.94**	**0.77**	**0.94**
seq02	0.34	0.71	0.34	0.76	0.48	0.86	0.60	0.88	0.60	0.88	**0.64**	**0.90**	**0.64**	**0.90**	**0.64**	**0.90**
seq03	0.39	0.76	0.40	0.77	0.60	0.89	0.72	0.89	0.73	0.89	0.75	**0.92**	**0.76**	**0.92**	0.75	**0.92**
seq04	0.54	0.76	0.56	0.77	0.71	0.87	0.65	0.88	0.67	0.88	**0.73**	**0.92**	**0.73**	**0.92**	**0.73**	**0.92**
seq05	0.37	0.82	0.38	0.82	0.67	0.94	0.83	0.94	0.83	0.95	**0.86**	**0.97**	**0.86**	0.96	**0.86**	**0.97**
seq06	0.46	0.81	0.48	0.82	0.52	0.89	0.66	0.89	0.68	0.90	0.75	0.94	**0.78**	**0.95**	0.75	0.94
seq07	0.44	0.69	0.44	[illegible]	0.65	0.85	0.72	0.89	0.70	0.89	**0.76**	**0.92**	**0.76**	**0.92**	**0.76**	**0.92**
seq08	0.32	0.62	0.28	0.60	0.34	0.74	0.46	0.78	0.49	0.78	**0.50**	0.80	**0.50**	**0.81**	**0.50**	0.80
seq09	0.28	0.42	0.29	0.43	0.58	0.72	0.70	0.80	0.71	0.80	**0.78**	**0.84**	**0.78**	**0.84**	**0.78**	**0.84**
seq10	0.47	0.69	0.48	0.70	0.62	0.85	0.66	0.89	0.67	0.89	**0.70**	**0.91**	**0.70**	**0.91**	**0.70**	**0.91**
seq11	0.57	0.64	0.58	0.65	0.72	0.76	0.78	0.81	0.81	0.82	0.81	**0.85**	**0.83**	**0.85**	0.81	**0.85**
seq12	0.27	0.57	0.28	0.58	0.54	0.75	0.59	0.80	0.60	0.80	**0.63**	**0.83**	0.62	**0.83**	**0.63**	**0.83**
seq13	0.41	0.74	0.42	0.74	0.53	0.89	0.60	0.91	0.63	0.92	**0.65**	**0.93**	**0.65**	**0.93**	**0.65**	**0.93**
seq14	0.55	0.84	0.56	0.84	0.73	0.90	0.77	0.89	0.75	0.89	**0.79**	**0.91**	0.78	**0.91**	**0.79**	**0.91**
seq15	0.46	0.68	0.46	0.68	0.51	0.84	0.64	0.87	0.62	0.87	**0.65**	0.89	**0.65**	**0.90**	**0.65**	0.89
seq16	0.50	0.69	0.52	0.70	0.65	0.83	0.68	0.85	0.70	0.85	0.73	**0.88**	**0.74**	**0.88**	0.73	**0.88**
seq17	0.65	0.78	0.66	0.80	0.75	**0.96**	0.74	0.95	0.76	0.95	0.78	**0.96**	**0.79**	**0.96**	0.78	**0.96**
seq18	0.53	0.74	0.55	0.75	0.66	0.90	0.74	0.92	0.75	0.92	**0.80**	**0.94**	0.79	**0.94**	**0.80**	**0.94**
seq19	0.48	0.63	0.49	0.64	0.53	0.68	0.51	0.66	0.53	0.67	**0.57**	**0.71**	**0.57**	**0.71**	**0.57**	**0.71**
seq20	0.20	0.55	0.21	0.56	0.52	0.80	0.81	0.89	0.82	0.90	0.82	**0.93**	**0.83**	**0.93**	0.82	**0.93**
seq21	0.31	0.59	0.32	0.60	0.51	0.84	0.64	0.90	0.63	0.90	**0.67**	**0.92**	**0.67**	**0.92**	**0.67**	**0.92**
seq22	0.48	0.81	0.48	0.82	0.67	0.94	0.81	0.95	0.81	0.95	**0.88**	**0.97**	**0.88**	**0.97**	**0.88**	**0.97**
seq23	0.37	0.73	0.38	0.74	0.52	0.87	0.57	0.87	0.59	0.88	0.62	**0.90**	**0.63**	**0.90**	0.62	**0.90**
seq24	0.34	0.78	0.36	0.79	0.82	0.95	0.92	0.96	0.92	0.96	**0.95**	**0.98**	**0.95**	**0.98**	**0.95**	**0.98**
seq25	0.47	0.74	0.48	0.75	0.79	0.90	0.76	0.89	0.77	0.90	**0.82**	**0.92**	**0.82**	**0.92**	**0.82**	**0.92**
seq26	0.21	0.58	0.22	0.59	0.64	0.84	0.74	0.91	0.74	0.91	0.78	**0.93**	**0.79**	**0.93**	0.78	**0.93**
seq27	0.26	0.61	0.26	0.61	0.63	0.87	0.83	0.95	0.83	0.95	**0.85**	**0.96**	**0.85**	**0.96**	**0.85**	**0.96**
seq28	0.24	0.56	0.27	0.57	0.47	0.78	0.74	0.89	0.75	0.89	**0.77**	**0.91**	**0.77**	**0.91**	**0.77**	**0.91**
seq29	0.37	0.67	0.38	0.67	0.50	0.81	0.60	0.85	0.59	0.85	**0.64**	**0.88**	0.62	**0.88**	**0.64**	**0.88**
seq30	0.20	0.49	0.21	0.50	0.33	0.72	0.62	0.85	0.62	0.85	**0.66**	**0.88**	**0.66**	**0.88**	**0.66**	**0.88**
seq31	0.41	0.50	0.42	0.51	0.53	0.59	0.55	0.63	0.56	0.63	**0.60**	**0.67**	0.58	**0.67**	**0.60**	**0.67**
seq33	0.29	0.68	0.30	0.69	0.52	0.83	0.61	0.87	0.61	0.87	**0.66**	**0.89**	**0.66**	**0.89**	**0.66**	**0.89**
seq34	0.32	0.79	0.33	0.80	0.73	0.94	0.87	0.96	0.87	0.96	**0.89**	**0.97**	**0.89**	**0.97**	**0.89**	**0.97**
seq35	0.34	0.72	0.35	0.73	0.50	0.87	0.54	0.89	0.54	0.89	**0.59**	**0.91**	0.58	**0.91**	**0.59**	**0.91**
seq36	0.40	0.75	0.40	0.76	0.51	0.88	0.56	0.89	0.56	0.89	0.58	**0.91**	**0.60**	**0.91**	0.58	**0.91**

Alternating solvers

There is a structure in the problem → we can do it more efficiently

1. Full formulation (**Intractable**)

$$\lambda_i \mathtt{x}_i = \lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = \mathtt{R}(r_i)\mathtt{X}_i + \mathtt{C}(r_i)$$

2. Rotation during the capture & initial linearized (**P3P initialization, Slow**)

$$\lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = (\mathtt{I} + (r_i - r_0)[\mathtt{w}]_\times)(\mathtt{I} + [\mathtt{v}]_\times)\mathtt{X}_i + \mathtt{C} + (r_i - r_0)\mathtt{t}$$

3. Rotation during the capture linearized (**No initialization needed, Slow**)

$$\lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = (\mathtt{I} + (r_i - r_0)[\mathtt{w}]_\times)\mathtt{R}_\mathtt{v}\mathtt{X}_i + \mathtt{C} + (r_i - r_0)\mathtt{t}$$

4. Alternating between 2 linear problems (**P3P initialization, FAST**) $\mathrm{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v},\mathtt{C},\mathtt{w},\mathtt{t}}$

$$\lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = (\mathtt{I} + (r_i - r_0)[\mathtt{w}]_\times)\mathtt{X}_i + [\mathtt{v}]_\times\mathtt{X}_i + (r_i - r_0)[\mathtt{w}]_\times[\hat{\mathtt{v}}]_\times\mathtt{X}_i + \mathtt{C} + (r_i - r_0)\mathtt{t}$$

Alternating solvers

There is a structure in the problem → we can do it more efficiently

2. Rotation during the capture & initial linearized (**P3P initialization, Slow**)

$$\lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = \underbrace{\left(\mathtt{I} + (r_i - r_0)[\mathtt{w}]_\times\right)\left(\mathtt{I} + [\mathtt{v}]_\times\right)\mathtt{X}_i} + \mathtt{C} + (r_i - r_0)\mathtt{t}$$

expand

4. Alternating between 2 linear problems

$\mathrm{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v},\mathtt{C},\mathtt{w},\mathtt{t}}$

$$\lambda_i \begin{bmatrix} r_i \\ c_i \\ 1 \end{bmatrix} = \overbrace{\left(\mathtt{I} + (r_i - r_0)[\mathtt{w}]_\times\right)\mathtt{X}_i + [\mathtt{v}]_\times\mathtt{X}_i + (r_i - r_0)\underbrace{[\mathtt{w}]_\times[\hat{\mathtt{v}}]_\times}_{\text{alternate}}\mathtt{X}_i} + \mathtt{C} + (r_i - r_0)\mathtt{t}$$

Fix : $[\hat{\mathtt{v}}]_\times$ compute $\mathtt{v}, \mathtt{C}, \mathtt{w}$ and $\mathtt{t}$

Fix $\mathtt{v}, \mathtt{C}, \mathtt{w}$ and $\mathtt{t}$ compute: $[\hat{\mathtt{v}}]_\times$

Alternating solvers

Simulated experiments … the green alternating solver as good as R6P

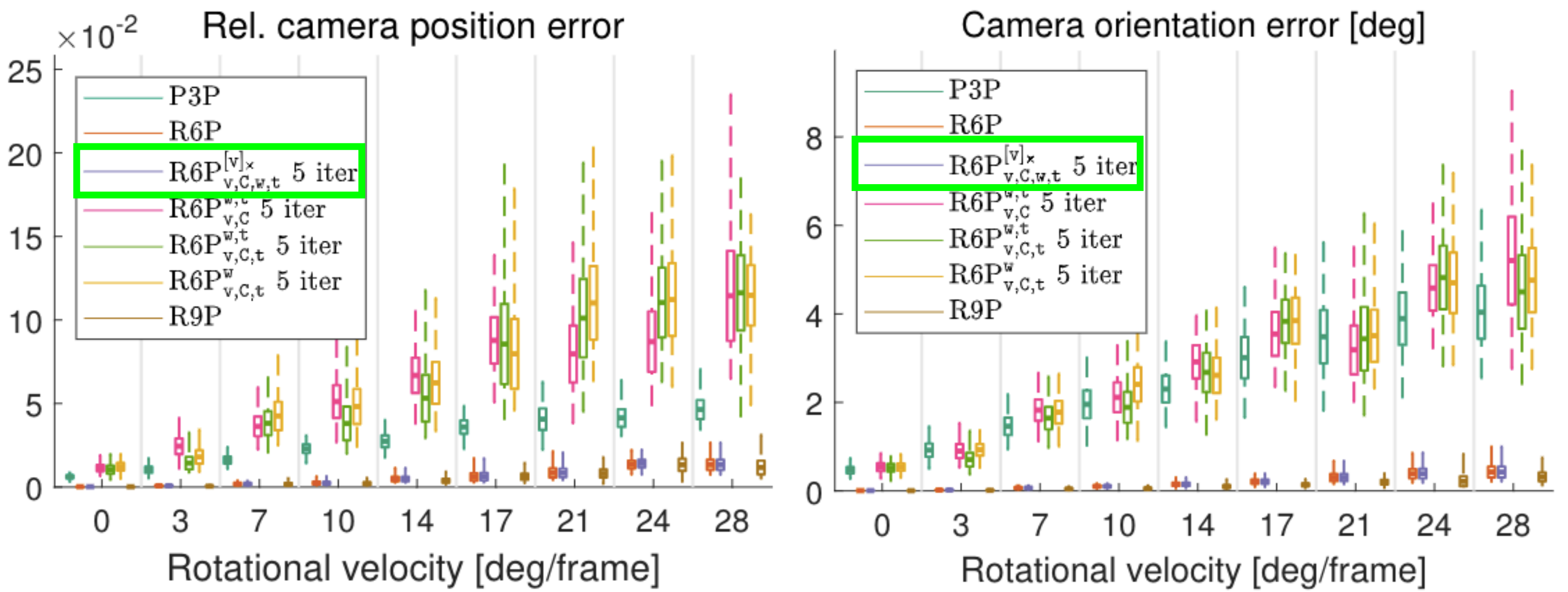
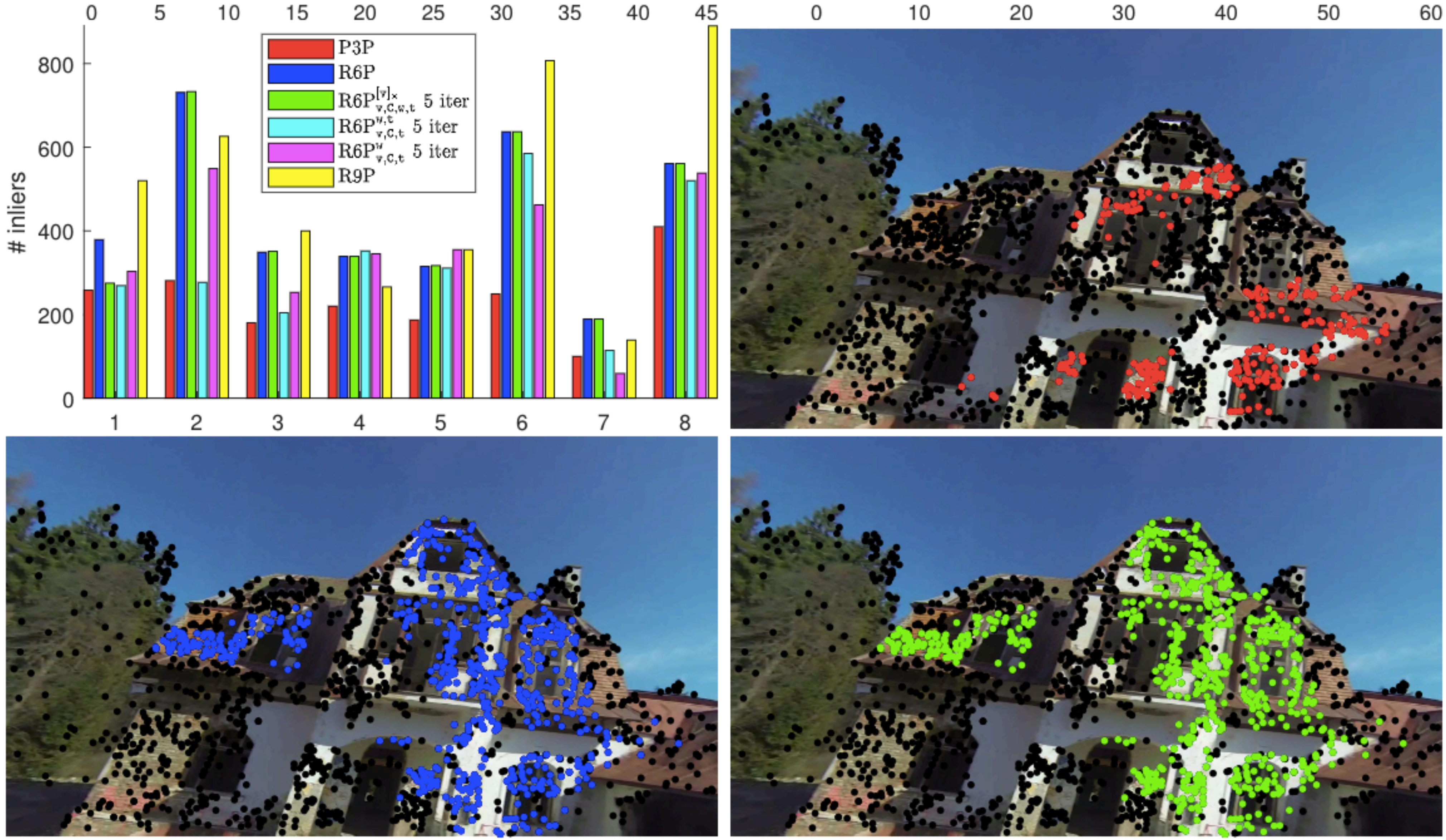


Fig. 5. Increasing the camera motion and estimating camera pose with all solvers being initialized with P3P. $R6P^{[v]_\times}_{v,C,w,t}$ and R9P now provide consistently excellent results, comparable or outperforming those of R6P at a fraction of the computation cost. $R6P^{w,t}_{v,C}$, $R6P^{w}_{v,C,t}$ and $R6P^{w,t}_{v,C,t}$ with 50 iterations now perform better than P3P, but still not as good as the other RS solvers.

Alternating solvers

Real experiments … the green alternating solver as good as red R6P



Alternating solvers

SPEED

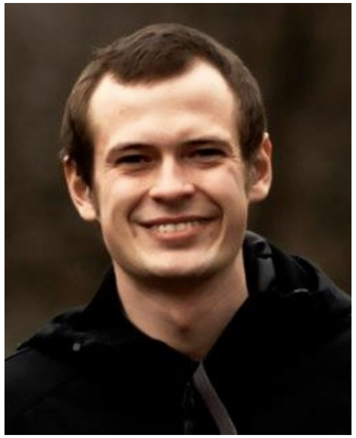
Table 1. Average timings on 2.5GHz i7 CPU per iteration for all used solvers.

solver	P3P	R6P	$\mathrm{R6P}^{[\mathtt{v}]_\times}_{\mathtt{v,C,w,t}}$	$\mathrm{R6P}^{\mathtt{w}}_{\mathtt{v,C,t}}$	$\mathrm{R6P}^{\mathtt{w,t}}_{\mathtt{v,C,t}}$	$\mathrm{R6P}^{\mathtt{w,t}}_{\mathtt{v,C}}$	R9P
time per iteration	$3\mu s$	$1700\mu s$	$10\mu s$	$24\mu s$	$30\mu s$	$27\mu s$	$20\mu s$
max # of solutions	4	20	1	1	1	1	1

Computing Rolling Shutter Camera Pose via optimized algebraic geometry

Tomas Pajdla

C. Albl, Z. Kukelova, V. Larsson, A. Sugimoto


CTU CZECH TECHNICAL UNIVERSITY IN PRAGUE

Czech Technical University in Prague

ETH Zürich

NII 大学共同利用機関法人 情報・システム研究機構 国立情報学研究所 National Institute of Informatics