Supplementary material
A sparse resultant based method for efficient minimal solvers

Snehal Bhayani[1] Zuzana Kukelova[2] Janne Heikkilä [1]

[1]Center for Machine Vision and Signal Analysis, University of Oulu, Finland
[2]Faculty of Electrical Engineering, Czech Technical University in Prague

1. Existing sparse resultant based algorithms

In this section we consider the existing sparse resultant based algorithms [4, 6], where the authors consider a system of n polynomials,

$$\{f_1(x_1, ..., x_n) = 0, ..., f_n(x_1, ..., x_n) = 0\}, \tag{1}$$

in n unknowns, $X = \{x_1, ..., x_n\}$ for computing a sparse resultant matrix. While Heikkilä [6] propose a method to hide one variable, Emiris [4] propose two methods, one where they hide a variable, and another where they add an extra polynomial of the form $u_0 + u_1x_1 + \cdots + u_nx_n$, for generating a polynomial solver. In each of these methods the underlying assumption is that there are as many polynomials as there are unknowns. Hence, using their proposed algorithms we could not generate solvers for those minimal problems with more polynomials than unknowns. Additionally, the algorithm by [6] suffers from other drawbacks as well:

1. Heikkilä [6] propose an method of hiding one variable, say x_n, and computing a monomial basis B to linearize the input polynomial equations to have

$$\mathtt{M}(x_n)\mathbf{b} = 0, \tag{2}$$

where $\mathbf{b} = \text{vec}(B)$ based on some monomial order. However such a monomial basis can lead to a coefficient matrix $\mathtt{M}(x_n)$ that is rank deficient and hence leads to unstable or incorrect solvers.

2. If in case $\mathtt{M}(x_n)$ is not rank deficient Heikkilä [6] transform (2) into a generalized eigenvalue problem(GEP) of the form

$$\mathtt{A}\mathbf{y} = x_n\mathtt{B}\mathbf{y}. \tag{3}$$

as described in (4) of Section 2.2 of our main paper. But such a conversion leads to large and sparse $\mathtt{A}$ and $\mathtt{B}$ that introduces parasitic eigenvalues which are either 0 or ∞. It can also lead to spurious eigenvalues that correspond to incorrect solutions.

1.1. Proposed extension to Heikkilä's algorithm

Considering the shortcomings of the method by Heikkilä [6] we attempted to extend and improve their algorithm,

1. Due to an iterative nature of the algorithm, it is easy to relax the requirement of having the same number of equations and unknowns, and hence we assume that there are $m \geq n$ polynomial equations with n unknowns. Then we perform an exhaustive search across all polynomial combinations and variables by hiding each variable $x_i \in X$ at a time. This usually reduces the size of the monomial basis leading to a smaller matrix $\mathtt{M}(x_n)$ than the one generated by Heikkilä's algorithm [6].

2. The problem of rank deficiency is resolved by testing for rank of the matrix $\mathtt{M}(x_n)$ for every prospective monomial basis B so chosen in the algorithm. This guarantees that the eigenvalues and eigenvectors of GEP formulation provides correct solutions to the original polynomial system (1).

3. Additionally, we know that a GEP formulation for many minimal problems in computer vision has parasitic zero(or ∞) eigenvalues due to zero columns in $\mathtt{A}$(or $\mathtt{B}$) in (3). Hence we extended the the algorithm by Heikkilä [6] to eliminate a set of rows-columns in order to reduce the size of GEP we are trying to solve.

The sizes of solvers generated using these extensions to the algorithm by Heikkilä [6] for some interesting minimal problems are listed in Table 1(Column 1). If in these solvers $\mathtt{A}$ or $\mathtt{B}$ in GEP (3) is an invertible matrix, GEP can be executed as a sequence of a matrix inverse and an eigendecomposition of the resulting matrix. For example, a GEP of size 18×18 means an inverse of 18×18 matrix and an eigenvalue decomposition of 18×18 matrix. We note that this assumption holds true for all of the minimal problems in Table 1. In such a case the most computationally expensive step is the eigenvalue decomposition, since the matrix that is inverted is usually sparse. Now, it can be seen that for most

Problem	Extension to [6]	Our		u-resultant	
	GEP	Inv	Eig.	Inv	Eig.
Rel. pose F+λ 8pt(8 sols.)	12×12	$\mathbf{11 \times 11}$	$\mathbf{9 \times 9}$	15×15	9×9
Stitching $f\lambda$+R+$f\lambda$ 3pt (18 sols.)	24×24	$\mathbf{18 \times 18}$	$\mathbf{18 \times 18}$	31×31	18×18
Rel. pose E+λ 6pt (26 sols.)	30×30	$\mathbf{14 \times 14}$	$\mathbf{26 \times 26}$	44×44	26×26
Abs. pose quivers (20 sols.)	43×43	$\mathbf{68 \times 68}$	$\mathbf{24 \times 24}$	-	-
Rel. pose f+E+f 6pt (15 sols.)	18×18	$\mathbf{12 \times 12}$	$\mathbf{18 \times 18}$	-	-
Rel. pose λ_1+F+λ_2 9pt (24 sols.)	68×68	$\mathbf{90 \times 90}$	$\mathbf{27 \times 27}$	-	-
Rel. pose E+$f\lambda$ 7pt (19 sols.)	36×36	$\mathbf{61 \times 61}$	$\mathbf{19 \times 19}$	105×105	19×19
Rel. pose λ+E+λ 6pt (52 sols.)	110×110	$\mathbf{39 \times 39}$	$\mathbf{56 \times 56}$	-	-
Triangulation from satellite im.(27 sols.)	52×52	$\mathbf{88 \times 88}$	$\mathbf{27 \times 27}$	93×93	27×27
Unsynch. Rel. pose (16 sols.)	128×128	$\mathbf{150 \times 150}$	$\mathbf{18 \times 18}$	-	-
Rolling shutter pose (8 sols.)	18×18	$\mathbf{47 \times 47}$	$\mathbf{8 \times 8}$	48×48	8×8

Table 1. A comparison of the sizes of important computation steps performed by solvers generated using our new method with that of the solvers generated based on our attempted extensions of the algorithm by Heikkilä [6] as well as the solvers generated using an u-resultant based method. Missing entry is for the case where we failed to generate a solver.

of the minimal problems our proposed solvers are solving substantially smaller eigenvalue problems than the solvers based on the extended version of [6]. And even though for few minimal problems the matrices to invert in our proposed solvers are slightly larger than the inverses in solvers based on [6], these matrices are usually quite sparse and the size difference is not as dominating as the difference in size of eigenvalue problem. Additionally, a GEP would lead to parasitic eigenvalues corresponding to incorrect solutions and extra computation has to be carried out in order to eliminate such eigenvalues, thus slowing down such solvers even further as compared to the ones based on our method. Additionally the number of eigenvalues to be computed for a GEP still is quite large as compared to the eigenvalues to be computed by our proposed solver. Hence based on these considerations, we can conclude that our proposed solvers for all of the problems in Table 1 would be faster than the ones generated using our proposed extensions to [6].

1.2. Comparison with Emiris's u-resultant method

Now we consider the u-resultant based method [4] where the authors add a polynomial of a general form $u_0 + x_1u_1 + \cdots + x_nu_n$ with random coefficients, to the original equation (4). However we note that in general the method presented in [4] does not work for a system with more polynomial equations than unknowns. Moreover, there is no publicly available code for the method [4]. Therefore, for a fair comparison with our method that based upon adding a polynomial of a special form, we modified our resultant-based method to simulate the one from [4]. For this, we augmented (4) with a polynomial of the form $u_0 + x_1u_1 + \cdots + x_nu_n$ by selecting $u_1, \ldots, u_n$ randomly from $\mathbb{Z}$ (for more details on u-resultant we refer to [4, 3]). The column 3 in Table 1 lists the sizes of solvers generated in this manner and is compared with the sizes of solvers generated based on our proposed method. We can observe that for many minimal problems the size of matrix to be inverted based on general u-resultant method is larger than that of the matrix to be inverted in our proposed solver. This indicates that our proposed solver would be faster than the solvers based on general u-resultant method for such minimal problems. Beyond this, for several minimal problems (5 problems from Table 1), we either failed to generate a working solver by using the above mentioned general u-polynomial at all or within a reasonable amount time by testing polynomial combinations of a reasonable size. We refer to Algorithm 1 here and Section 3.2 of our main paper for more details about the iterative nature adopted for testing polynomial combinations of various sizes.

Additionally we also considered the problem from computational biology explored in [4]. We compare the size of the u-resultant based solver for this problem reported in [4], with the size of a solver generated using our proposed method. This problem consists of 3 polynomial equations in 3 variables with 15 generic coefficients. For more details of the algebraic problem formulation, we refer to Section 7 in [4]. Now, the mixed volume of the input polynomial system is 16 which denotes the actual number of solutions to this polynomial system. The solver considered in [4] is generated using the u-resultant method by adding an extra polynomial of the form, $f_0 = u + 31x_1 - 41x_2 + 61x_3$. The solver consists of an inverse of matrix of size 56×56 and an eigenvalue decomposition of 30×30 matrix. We generated a solver for the same algebraic formulation with our proposed algorithm. Our new solver includes a matrix inversion of smaller matix of size 48×48 as well as smaller eigenvalue problem of size 16×16. This shows that the solver generated using our proposed algorithm would be faster than the one considered in [4].

2. Algorithms

Now we consider the main contribution of our main paper for which we described a three step procedure that leads to an eigenvalue formulation(Equations (14) or (16) in our main paper) to be solved for extracting roots to (4). So here we provide algorithms for each of these three steps. For the sake of this section, we assume details and notations of Section 3 of our main paper. We also consider a set of monomial multiples T to be of form $\{T_1, \ldots, T_m\}$ where each T_i represents the set of monomial multiples for polynomial $f_i(x_1, \ldots, x_n)$. Additionally, we shall assume that wherever required a coefficient matrix $\mathtt{M}$ is computed from a basis B along with a corresponding set of monomial multiples T, following the lines of Section 3. With these details in mind, we now outline Algorithm 1 for computing a monomial B basis from a set of m polynomial equations,

$$\{f_1(x_1, \ldots, x_n) = 0, \ldots, f_m(x_1, \ldots, x_n) = 0\} \qquad (4)$$

in n variables. The output of the algorithm also contains a set of monomial multiples, T as well as the coefficient matrix computed from B and T. For details about the underlying theory, we refer to Section 3.1 in our main paper. For an alternate eigenvalue formulation(Equation (16) in our main paper), we need to change Step 14 in Algorithm 1 to $B'_\lambda \leftarrow \{\mathbf{x}^m \in T'_{m+1} \mid x_i\mathbf{x}^m \in B'\}$, $B'_c \leftarrow B' - B'_\lambda$.

2.1. Removing columns from $\mathtt{M}$

The next step in our proposed method is to reduce the monomial basis B by removing columns from $\mathtt{M}$ along with a corresponding set of rows. A brief procedure for this step is described in Section 3.3 of our main paper, while the Algorithm 2, listed here achieves this. The input is the monomial basis B and the set of monomial multiples T computed by Algorithm 1 and the output is a reduced monomial basis B_{red} and a reduced set of monomial multiples, T_{red} that index the columns and rows of the reduced matrix $\mathtt{M}_{\text{red}}$ respectively. We note that this algorithm is the same irrespective of the version of eigenvalue formulation to be considered(Equations (14) or (15) in our main paper).

Now, it may happen that the reduced matrix $\mathtt{M}_{\text{red}}$ still has more rows than columns. Hence in our main paper, we have outlined an idea to remove excess rows so as to transform $\mathtt{M}_{\text{red}}$ into a square matrix to facilitate a decomposition of resultant matrix constraint to an eigenvalue formulation of equation (14)(or the alternate eigenvalue formulation of equation (16). For more details we refer to Proposition 3.1 in our main paper). Towards this we provide Algorithm 3 to remove the extra rows from $\mathtt{M}_{\text{red}}$ by removing some monomial multiples from T_{red}. It accepts B_{red} and T_{red} as input and returns a set of monomial multiples, T_{sq} that along with the basis B_{red}, leads to square matrix $\mathtt{M}_{\text{sq}}$. For an alternate eigenvalue formulation(Equation (16) in our main paper), we just need to change Step 16 in Algorithm 3 to $B'_\lambda \leftarrow \{\mathbf{x}^m \in T'_{m+1} \mid x_i\mathbf{x}^m \in B'\}$, $B'_c \leftarrow B' - B'_\lambda$.

Algorithm 1 Extracting favourable monomial basis using extra equation

Input $F = \{f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\}$, $\mathbf{x} = [x_1, \ldots, x_n]$
Output $B, T, \mathtt{M}$

1: $B \leftarrow \phi, T \leftarrow \phi$
2: **for** $i \in \{1, \ldots, n\}$ **do**
3: $F' \leftarrow \{f_1, \ldots, f_{m+1}\}$, $f_{m+1} = x_i - \lambda$
4: Calculate the support of the input polynomials: $A_j \leftarrow \text{supp}(f_j), j = 1, \ldots, m+1$
5: Construct newton polytopes: $NP_j \leftarrow \text{conv}(A_j), j = 1, \ldots, m+1$ as well as a unit simplex $NP_0 \subset \mathbb{Z}^n$.
6: Enumerate combinations of indices of all possible sizes: $K \leftarrow \{\{k_0, \ldots, k_i\} \mid \forall 0 \leq i \leq (m+1); k_0, \ldots, k_i \in \{0, \ldots, m+1\}; k_j < k_{j+1}\}$
7: Let $\Delta \leftarrow \{\{\delta_1, \ldots, \delta_{n+1}\} \mid \delta_i \in \{-\epsilon, 0, \epsilon\}; i = 1, \ldots, (n+1)\}$ denote the set of possible displacement vectors
8: **for** $I \in K$ **do**
9: Compute the minkowski sum, $Q \leftarrow \sum_{j \in I}(NP_j)$
10: **for** $\delta \in \Delta$ **do**
11: $B' \leftarrow \mathbb{Z}^n \cap (Q + \delta)$
12: $T'_j \leftarrow \{t \in \mathbb{Z}^n \mid t + A_j \subset B'\}, j = 1 \ldots m+1$
13: $T' \leftarrow \{T'_1 \ldots T'_{m+1}\}$
14: $B'_\lambda \leftarrow B' \cap T'_{m+1}$, $B'_c \leftarrow B' - B'_\lambda$
15: Compute $\mathtt{M}'$ from B' and T'
16: **if** $\Sigma_{j=1}^{m+1}|T'_j| \geq |B'|$ and $\min_j |T'_j| > 0$ and $rank(\mathtt{M}') = |B'|$ **then**
17: $\mathtt{A}_{12} \leftarrow$ submatrix of $\mathtt{M}'$ column indexed by B'_c and row indexed by $T'_1 \cup \cdots \cup T'_m$
18: **if** $\text{rank}(\mathtt{A}_{12}) = |B'_c|$ and $|B| \geq |B'|$ **then**
19: $B \leftarrow B', T \leftarrow T'$
20: **end if**
21: **end if**
22: **end for**
23: **end for**
24: **end for**
25: Compute $\mathtt{M}$ from B and T

3. Experiments

In Table 2 we provide a comparison of solvers' sizes for some additional interesting minimal problems. We can see from the table, that for all considered minimal problems our proposed method generates the smallest solvers (sometimes of the same size as Gröbner basis solvers generated with methods from [11, 13]). For an interpretation of the solver sizes, we refer to Section 4.1 of Evaluation in our main

Algorithm 2 Reducing the monomial basis

Input: B, T
Output: $B_{\text{red}}, T_{\text{red}}, \mathtt{M}_{\text{red}}$

$B' \leftarrow B, T' \leftarrow T$
repeat
  stopflag $\leftarrow$ True
  Compute $\mathtt{M}'$ from B' and T'
  for column c in $\mathtt{M}'$ **do**
    Copy $\mathtt{M}'$ to $\mathtt{M}''$
    Remove rows $r_1, \ldots, r_k$ containing c from $\mathtt{M}''$
    Remove columns $c_1, \ldots, c_l$ of $\mathtt{M}''$ present in $r_1, \ldots, r_k$
    if $\mathtt{M}''$ satisfies Proposition 3.1 **then**
      Remove monomials from B' indexing columns $c_1, \ldots, c_l$
      Remove monomials from T' indexing rows $r_1, \ldots, r_k$
      stopflag $\leftarrow$ False
      break
    end if
  end for
until stopflag is True
$B_{\text{red}} \leftarrow B', T_{\text{red}} \leftarrow T'$
Compute $\mathtt{M}_{\text{red}}$ from B_{red} and T_{red}

Algorithm 3 Removal of excess rows

Input $B_{\text{red}}, T_{\text{red}}$
Output $T_{\text{sq}}, \mathtt{M}_{\text{sq}}$

T_{red} contains $\{T'_1, \ldots, T'_{m+1}\}$
$B_N \leftarrow |B_{\text{red}}|, T_N \leftarrow \Sigma_{j=1}^{m+1} |T'_j|, t_{\text{chk}} \leftarrow \phi$
while $T_N > B_N$ **do**
  $B' \leftarrow B_{\text{red}}, T' \leftarrow T_{\text{red}}$
  T' contains $\{T'_1, \ldots, T'_{m+1}\}$
  Randomly select $t \in \{t_m \in T'_{m+1} \mid (t_m, m+1) \notin t_{\text{chk}}\}$
  if t **then**
    $T'_{m+1} \leftarrow T'_{m+1} - \{t\}, T' \leftarrow \{T'_1, \ldots, T'_{m+1}\}$
    $t_{\text{chk}} \leftarrow t_{\text{chk}} \cup \{(t, m+1)\}$
  else
    Randomly select $i \in \{1, \ldots, m\}$
    Randomly select $t \in \{t_i \in T'_i \mid (t_i, i) \notin t_{\text{chk}}\}$
    $T'_i \leftarrow T'_i - \{t\}, T' \leftarrow \{T'_1, \ldots, T'_{m+1}\}$
    $t_{\text{chk}} \leftarrow t_{\text{chk}} \cup \{(t, i)\}$
  end if
  $B'_\lambda \leftarrow B' \cap T'_{m+1}, B'_c \leftarrow B' - B'_\lambda$
  Compute $\mathtt{M}'$ from B' and T'
  if $\min_j |T'_j| > 0$ and $\mathit{rank}(\mathtt{M}') = |B'|$ **then**
    $\mathtt{A}_{12} \leftarrow$ submatrix of $\mathtt{M}'$ column indexed by B'_c and row indexed by $T'_1 \cup \cdots \cup T'_m$
    if $\text{rank}(\mathtt{A}_{12}) = |B'_c|$ **then**
      $T_{\text{red}} \leftarrow T', T_N \leftarrow \Sigma_{j=1}^{m+1} |T'_j|$
    end if
  end if
end while
$T_{\text{sq}} \leftarrow T_{\text{red}}$
Compute $\mathtt{M}_{\text{sq}}$ from B_{red} and T_{sq}

paper. We also note that, for two of the problems in Table 2, we failed to generate a solver using the Gröbner fan method [13] in a reasonable amount of time.

Table 3 performs a stability comparison of the solvers for minimal problems from Table 2 as well as for the problems from our main paper that were considered for comparison of sizes but were left out from the stability comparison due to the lack of space in the main paper. Just as in our main paper we measure the mean and median of Log_{10} of the normalized equation residuals for computed solutions as well as the solvers failures as a % of 5K instances for which at least one solution has a normalized residual $> 10^{-3}$. Then our observation from the stability comparisons in Table 2 of the main paper is corroborated with our observations here for these extra set of minimal problems in Table 3. We notice that here as well, most of the solvers based on our proposed method are similarly or more stable than the ones based on Gröbner basis methods [11, 13] and with less failures.

Figure 1 shows histogram of Log_{10} of normalized equation residuals for the "Rel.pose λ+E+λ" problem whose stability was compared in Table 2 of our main paper. We note that our solver is not only faster, but also more stable than the state-of-the-art solvers. Additionally we provide histograms of residuals in Figure 3 for other interesting minimal problems whose stability comparisons have also been performed either in Table 3 here or in the Table 2 of our main paper. The residuals have been obtained based on 5K runs on random input data points. We observe from these histograms that our proposed solvers have comparable stability w.r.t. the state-of-the-art solvers based on Gröbner basis [11] and heuristic-based solvers [13]. However an important measure of stability for real world applications is the % of failures of a minimal solver. Here, we have measured a solver's failure as the number of instances with large values of the equation residual(say above 10^{-3}) for computed solutions. Using this failure metric, we observe that our proposed resultant-based solvers for the four problems, Unsynch. Rel. pose [2], Rel. pose λ_1+F+λ_2 9pt [9], Optimal PnP (Cayley) [14] and Abs. pose refractive P5P [5] clearly have less failures than the state-of-the-art Gröbner basis and heuristic-based solvers. We also note that for four problems from Figure 3, i.e. Rel. pose f+E+f 6pt [9], Abs. pose refractive P5P [5], Rel. pose E+$f\lambda$ 7pt [8] and Optimal pose 2pt v2 [16], our proposed solvers are smaller than the state-of-the-art solvers based on Gröbner basis [11] and heuristic-based solvers [13]. Moreover, we note from the Table 1 of

Problem	Our	Original*	[11]	GFan [13]	(#GB)	Heuristic [13]
Rolling shutter pose (8 sols.)	$\mathbf{47 \times 55}$	48×56 [15]	$\mathbf{47 \times 55}$	$\mathbf{47 \times 55}$	(520)	$\mathbf{47 \times 55}$
Triangulation from satellite im. (27 sols.)	$\mathbf{87 \times 114}$	93×120 [17]	88×115	88×115	(837)	88×115
Optimal pose 2pt v2 (24 sols.)	$\mathbf{176 \times 200}$	192×216[16]	192×216	–	?	192×216
Optimal PnP (Cayley) (40 sols.)	$\mathbf{118 \times 158}$	$\mathbf{118 \times 158}$ [14]	$\mathbf{118 \times 158}$	$\mathbf{118 \times 158}$	(2244)	$\mathbf{118 \times 158}$
Optimal PnP (Hesch) (27 sols.)	$\mathbf{87 \times 114}$	93×120 [7]	88×115	88×115	(837)	88×115

Table 2. Comparison of sizes of solvers for some more minimal problems. Missing entries are when we failed to generate a Gröbner fan solver in reasonable time. $(*)$: Sizes for the original formulations.

Problem	Our			[11]			Heuristic [13]		
	mean	med.	fail(%)	mean	med.	fail(%)	mean	med.	fail(%)
Rel. pose F+λ 8pt	-14.26	-14.43	$\mathbf{0}$	-13.74	-14.26	0.14	-14.18	-14.48	$\mathbf{0}$
Rel. pose E+f 6pt	-13.17	-13.44	$\mathbf{0}$	-12.87	-13.17	0	-13.05	-13.34	$\mathbf{0}$
Rel. pose E+λ 6pt	-11.65	-11.94	$\mathbf{0.34}$	-11.42	-11.72	0.52	-11.34	-11.68	0.94
Stitching $f\lambda$+R+$f\lambda$ 3pt	-13.22	-13.42	$\mathbf{0}$	-13.06	-13.37	0.16	-13.20	-13.46	0.02
Rel. pose λ_1+F+λ_2 9pt	-9.81	-10.08	$\mathbf{3.32}$	-9.81	-10.39	5.14	-9.56	-9.98	6.10
Rel. pose E+$f\lambda$ 7pt (elim. $f\lambda$)	-10.71	-10.95	0.38	-10.57	-10.90	$\mathbf{0.30}$	-11.04	-11.32	0.32
Abs. pose quivers$^{(\dagger)}$	-12.39	-12.60	$\mathbf{0}$	-11.18	-11.51	0.32	-12.48	-12.88	$\mathbf{0}$
Rolling shutter pose	-12.16	-12.34	$\mathbf{0}$	-12.52	-12.72	$\mathbf{0}$	-12.43	-12.65	$\mathbf{0}$
Triangulation from satellite im.	-11.67	-11.80	$\mathbf{0}$	-11.53	-11.83	0.76	-11.61	-11.93	0.5
Optimal pose 2pt v2	-9.85	-10.04	$\mathbf{0.1}$	-10.85	-10.83	$\mathbf{0.1}$	-10.36	-10.61	$\mathbf{0.1}$
Optimal PnP (Cayley)	-9.14	-9.45	$\mathbf{3.64}$	-8.38	-8.74	10.28	-8.42	-8.75	7.64
Optimal PnP (Hesch)	-11.07	-11.34	0.98	-11.36	-11.72	0.82	-11.05	-11.36	$\mathbf{0.1}$
Unsynch. Rel. pose$^{(\ddagger)}$	-10.26	-10.40	$\mathbf{0}$	-8.13	-8.64	3.84	-9.93	-10.19	0.86

Table 3. A comparison of stability for solvers generated by our proposed resultant-based method, solvers generated using [11] and heuristic-based solvers [13] on some more minimal problems. Mean and median are computed from Log_{10} of normalized equation residuals. Missing entries are when we failed to extract solutions to all variables. $(\dagger)$: Input polynomials were eliminated using G-J elimination before generating a solver using our resultant method as well as solvers based on [11] and the heuristic-based solver [13]. $(\ddagger)$: Alternate eigenvalue formulation used for generating the solver based on our proposed method(see Proposition 3.1 in our main paper).

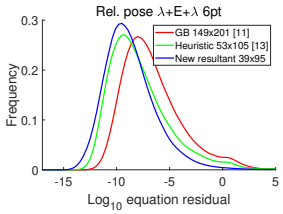

Figure 1. Histogram of Log_{10} normalized equation residual error for Rel. pose λ + E + λ 6pt problem

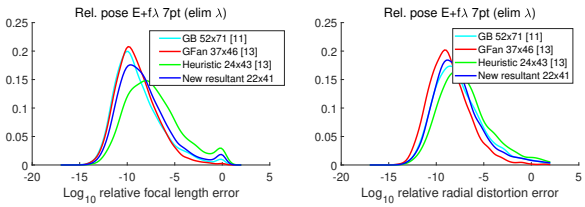

Figure 2. Histograms of (left) Log_{10} relative error in focal length, (right) Log_{10} relative error in radial distortion for Rel. pose E+$f\lambda$ 7pt (elimλ) problem.

the main paper, our proposed solver for the problem of Unsynch. Rel. pose [2], is significantly smaller than the competitive solvers for the same formulation of the problem.

E+$f\lambda$ solver on synthetic scenes: Here we show results from the synthetic experiment presented in the main paper. We studied the numerical stability of the new resultant-based solver for the problem of estimating the relative pose of one calibrated and one camera with unknown focal length and radial distortion from 7-point correspondences, i.e. the Rel. pose E+$f\lambda$ 7pt problem. We considered the formulation "elim. λ" proposed in [13] that leads to the smallest solvers. We studied the stability on 10K synthetically generated scenes as described in the main paper, see Section 4.1.

Figure 2 (left) shows Log_{10} of the relative error of the focal length f obtained by selecting the real root closest to the ground truth f_{gt} while (right) shows Log_{10} of the relative error of the distortion parameter λ obtained by selecting the real root closest to the ground truth λ_{gt}. All tested solvers provide stable results with only a small number of runs with

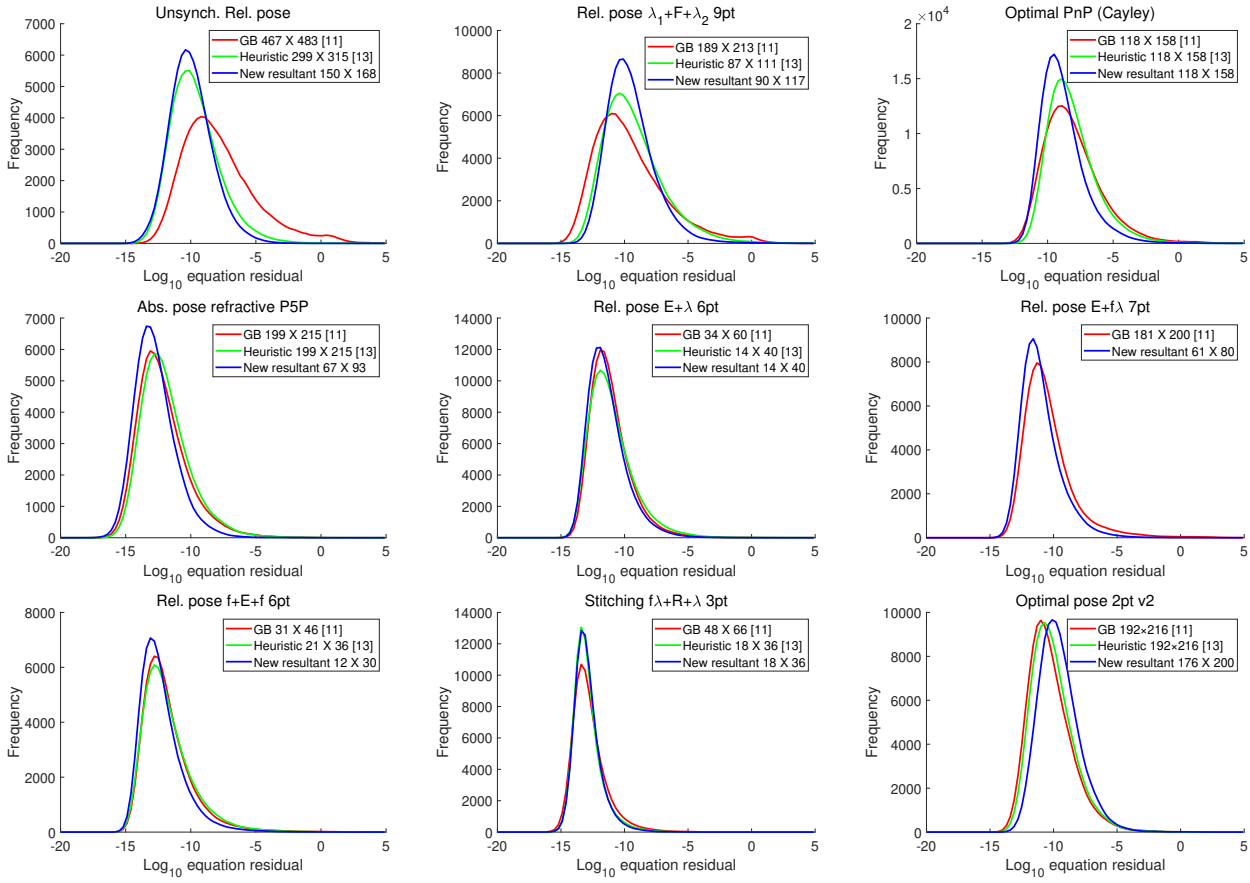

Figure 3. Histograms of Log_{10} of normalized equation residual error for nine selected minimal problems.

larger errors. The new resultant-based solver (blue) is not only smaller but also slightly more stable than the heuristic-based solver from [13] (green).

P4Pfr solver on real images: Here we show additional statistics for the real experiment presented in our main paper where we evaluated our proposed solver for the problem of estimating the absolute pose of a camera with unknown focal length and radial distortion from four 2D-to-3D point correspondences, i.e. the P4Pfr solver. We consider the *Rotunda* dataset, which was proposed in [10] and in [12] it was used for evaluating P4Pfr solvers. This dataset consists of 62 images captured by a GoPro Hero4 camera with significant radial distortion. The Rotunda reconstruction contains 170994 3D points and the average reprojection error was 1.4694 pixels over 549478 image points. We used the 3D model to estimate the pose of each image using our new P4Pfr resultant-based solver (28×40) in a RANSAC framework. Similar to [12], we used the camera and distortion parameters obtained from [1] as ground truth for the experiment.

In Table 4 we present the errors for the focal length, radial distortion, and the camera pose obtained using our proposed solver and for the sake of comparison we also list the errors, which were reported in [12], where the P4Pfr (40x50) solver was tested on the same dataset. Overall the errors are quite small, *e.g.* most of the focal lengths are within 0.1% of the ground truth and almost all rotation errors are less than 0.1 degrees, which shows that our new solver as well as the original solver work well for real data. The results of both solvers are very similar. However, we do take note that the slightly different values of errors are mainly due to RANSAC's random nature.

References

[1] RealityCapture. www.capturingreality.com. 6

[2] Cenek Albl, Zuzana Kukelova, Andrew W. Fitzgibbon, Jan Heller, Matej Smíd, and Tomás Pajdla. On the two-view geometry of unsynchronized cameras. *CoRR*, abs/1704.06843, 2017. 4, 5

[3] David A Cox, John Little, and Donal O'Shea. *Using algebraic geometry*, volume 185 of *Graduate Texts in Mathematics*. Springer-Verlag New York, 2005. 2

[4] Ioannis Z. Emiris. A general solver based on sparse resultants. *CoRR*, abs/1201.5810, 2012. 1, 2

Solver	Our P4Pfr 28×40			P4Pfr 40×50 [12]		
	avg.	med.	max	avg.	med.	max
Focal (%)	0.080	0.063	0.266	0.08	0.07	0.29
Distortion (%)	0.522	0.453	1.651	0.51	0.45	1.85
Rotation (degree)	0.031	0.029	0.062	0.03	0.03	0.10
Translation (%)	0.066	0.051	0.210	0.07	0.07	0.26

Table 4. Errors for the real Rotunda dataset. The errors are relative to the ground truth for all except rotation which is shown in degrees. The results for the P4Pfr solver (40×50) [12] are taken from [12]

[5] Sebastian Haner and Kalle Åström. Absolute pose for cameras under flat refractive interfaces. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1428–1436, 2015. 4

[6] Janne Heikkilä. Using sparse elimination for solving minimal problems in computer vision. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 76–84, 2017. 1, 2

[7] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (dls) method for pnp. In *2011 International Conference on Computer Vision*, pages 383–390, Nov 2011. 5

[8] Yubin Kuang, Jan Erik Solem, Fredrik Kahl, and Kalle Åström. Minimal solvers for relative pose with a single unknown radial distortion. In *Computer Vision and Pattern Recognition (CVPR)*, pages 33–40. IEEE, 2014. 4

[9] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. In *European Conference on Computer Vision (ECCV 2008), Proceedings, Part III*, volume 5304 of *Lecture Notes in Computer Science*, 2008. 4

[10] Zuzana Kukelova, Jan Heller, Martin Bujnak, Andrew Fitzgibbon, and Tomas Pajdla. Efficient solution to the epipolar geometry for radially distorted cameras. In *Proceedings of the IEEE international conference on computer vision*, pages 2309–2317, 2015. 6

[11] Viktor Larsson, Kalle Åström, and Magnus Oskarsson. Efficient solvers for minimal problems by syzygy-based reduction. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 3, 4, 5

[12] Viktor Larsson, Zuzana Kukelova, and Yinqiang Zheng. Making minimal solvers for absolute pose estimation compact and robust. In *International Conference on Computer Vision (ICCV)*, 2017. 6, 7

[13] Viktor Larsson, Magnus Oskarsson, Kalle Åström, Alge Wallis, Zuzana Kukelova, and Tomás Pajdla. Beyond grobner bases: Basis selection for minimal solvers. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3945–3954, 2018. 3, 4, 5, 6

[14] Gaku Nakano. Globally optimal DLS method for pnp problem with cayley parameterization. In *Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015*, pages 78.1–78.11, 2015. 4, 5

[15] Olivier Saurer, Marc Pollefeys, and Gim Hee Lee. A minimal solution to the rolling shutter pose estimation problem. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1328–1334. IEEE, 2015. 5

[16] Linus Svärm, Olof Enqvist, Fredrik Kahl, and Magnus Oskarsson. City-scale localization for cameras with known vertical direction. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1455–1461, 2017. 4, 5

[17] E. Zheng, K. Wang, E. Dunn, and J. Frahm. Minimal solvers for 3d geometry from satellite imagery. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 738–746, Dec 2015. 5