Supplementary Material
Solving Minimal Problems Without Matrix Inversion Using FFT-Based Interpolation

Haidong Wu Snehal Bhayani Janne Heikkilä
Center for Machine Vision and Signal Analysis
University of Oulu, Oulu, Finland
Haidong.Wu@oulu.fi, Snehal.Bhayani@oulu.fi, Janne.Heikkila@oulu.fi

S1. Resultant matrix sizes and runtime statistics

Table S1 summarizes the resultant matrix sizes used in our solver for all tested minimal problems, along with the average runtime per instance and the number of roots returned. These sizes correspond to the dimensions of the matrices $\mathrm{M}(x_1)$ constructed from the hidden variable formulation described in the main paper. As discussed in Section 4, the runtime of our solver is influenced primarily by the size of the resultant matrix and the number of roots that require evaluation and back-substitution.

The observations indicate that for problems with a similar number of roots, the runtime is largely determined by the size of the resultant matrix. For example, our solver returns 24 roots for the Rel. pose E+$f\lambda$ 7pt, Abs. pose refractive P5P, and Abs. pose quivers problems. However, the latter two are substantially slower because their resultant matrices are significantly larger (40×40) compared with that of the Rel. pose E+$f\lambda$ 7pt problem (20×20). Likewise, our solver returns a comparable number of roots for the Rel. pose f+E+f 6pt problem and the Abs. pose P4Pfr (elim. f) problem, but the latter exhibits a longer runtime because its resultant matrix is larger (20×20 vs. 10×10). Overall, these comparisons demonstrate that the size of the resultant matrix is the dominant factor governing the computational cost of our solver.

S2. Runtime breakdown of the proposed solver

Table S2 shows the runtime breakdown of the proposed solver for all tested minimal problems. The reported values correspond to the fraction of the total runtime spent in determinant coefficient recovery, polynomial root solving, and recovering remaining variables. The results indicate that determinant coefficient recovery and recovering remaining variables together dominate the overall computation. In many cases, recovering remaining variables constitutes the largest portion of the runtime. In contrast, polynomial root solving consistently requires only a small fraction of the total runtime. Other steps incur negligible runtime and are therefore omitted.

Algorithm S1 Offline Stage

Input: Input polynomial system from a minimal problem
Output: Hidden variable x_1; resultant matrix $\mathtt{M}(x_1)$; degree k of $\det(\mathtt{M}(x_1))$; number r of real solutions; valid row-column pair (i, j); and index pairs (j_1, j_2) for unhidden variables

Determine x_1, construct $\mathtt{M}(x_1)$, and compute the degree k using the sparse resultant method [1].
Randomize input coefficients and compute r using a Gröbner-basis.
Find a valid row-column pair (i, j) as follows:
for $i = 1$ to N **do**
  for $j = 1$ to N **do**
    Remove row i and column j from $\mathtt{M}(x_1)$ to obtain submatrix $\mathtt{M}^{(i,j)}(x_1)$
    Let G be $\gcd(\det(\mathtt{M}(x_1)), \det(\mathtt{M}^{(i,j)}(x_1)))$
    if G is a constant **then**
      Save (i, j) as a valid row-column pair
    end if
  end for
end for
Find index pairs (j_1, j_2) for each unhidden variable x_i $(i \geq 2)$:
for x_i in $\{x_2, \ldots, x_n\}$ **do**
  Find (j_1, j_2) such that $\deg(\mathbf{v}'[j_1], x_i) = \deg(\mathbf{v}'[j_2], x_i) + 1$
  if $\mathbf{v}'[j_1]$ and $\mathbf{v}'[j_2]$ differ only in x_i **then**
    Save (j_1, j_2) for computing the remaining variables, and continue to the next x_i.
  end if
end for

Table S1. Matrix sizes, runtimes, and root counts of our solver across different minimal problems. Values marked with * denote results before filtering spurious roots.

#	Problem	Our		
		time (ms)	roots	matrix size
1	Rel. pose F+λ 8pt (8 sols)	0.086	8	4×4
2	Rel. pose E+f 6pt (9 sols)	0.154	9	7×7
3	Rel. pose f+E+f 6pt (15 sols)	0.287	15	10×10
4	Stitching $f\lambda$+R+$f\lambda$ 3pt (18 sols)	0.213	18	6×6
5	Rel. pose E+$f\lambda$ 7pt (elim. λ) (19 sols)	0.417	19	35×35
6	Triangulation from satellite im. (27 sols)	2.923*	27	34×34
7	Optimal PnP (Hesch) (27 sols)	2.773*	27	33×33
8	Rolling shutter pose (8 sols)	1.526*	11	35×35
9	Abs. Pose P4Pfr (elim. f) (12 sols)	0.773*	14	20×20
10	Rel. pose E+$f\lambda$ 7pt (19 sols)	1.254*	24	20×20
11	Abs. pose refractive P5P (16 sols)	5.863*	24	40×40
12	Abs. pose quivers (20 sols)	3.221*	24	40×40
13	Optimal PnP (Cayley) (40 sols)	7.442*	43	45×45
14	Rel. pose λ_1+F+λ_2 9pt (24 sols)	9.520*	36	45×45

Table S2. Runtime breakdown of the proposed solver. Each value indicates the fraction of total runtime spent in the corresponding step.

#	Problem	Our		
		Determinant recovery	Polynomial root solving	Recovering remaining variables
1	Rel. pose F+λ 8pt	0.303	0.213	0.484
2	Rel. pose E+f 6pt	0.462	0.126	0.412
3	Rel. pose f+E+f 6pt	0.454	0.137	0.409
4	Stitching $f\lambda$+R+$f\lambda$ 3pt	0.148	0.152	0.700
5	Rel. pose E+$f\lambda$ 7pt (elim. λ)	0.458	0.120	0.422
6	Triangulation from satellite im.	0.428	0.036	0.536
7	Optimal PnP (Hesch)	0.434	0.042	0.524
8	Rolling shutter pose	0.390	0.023	0.587
9	Abs. Pose P4Pfr (elim. f)	0.393	0.050	0.557
10	Rel. pose E+$f\lambda$ 7pt	0.371	0.067	0.562
11	Abs. pose refractive P5P	0.358	0.021	0.621
12	Abs. pose quivers	0.315	0.027	0.658
13	Optimal PnP (Cayley)	0.490	0.030	0.480
14	Rel. pose λ_1+F+λ_2 9pt	0.478	0.024	0.498

References

[1] Snehal Bhayani, Zuzana Kukelova, and Janne Heikkilä. Computing stable resultant-based minimal solvers by hiding a variable. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6104–6111. IEEE, 2021. 1