GPU-Based Homotopy Continuation for Minimal Problems in Computer Vision

Chiang-Heng Chien
School of Engineering
Brown University
chiang-heng_chien@brown.edu

Hongyi Fan
School of Engineering
Brown University
hongyi_fan@brown.edu

Ahmad Abdelfattah
Innovative Computing Laboratory
University of Tennessee
ahmad@icl.utk.edu

Elias Tsigaridas
INRIA
elias.tsigaridas@inria.fr

Stanimire Tomov
Innovative Computing Laboratory
University of Tennessee
tomov@icl.utk.edu

Benjamin Kimia
School of Engineering
Brown University
benjamin_kimia@brown.edu

Abstract

Systems of polynomial equations arise frequently in computer vision, especially in multiview geometry problems. Traditional methods for solving these systems typically aim to eliminate variables to reach a univariate polynomial, e.g., *a tenth-order polynomial for 5-point pose estimation, using clever manipulations, or more generally using Grobner basis, resultants, and elimination templates, leading to successful algorithms for multiview geometry and other problems. However, these methods do not work when the problem is complex and when they do, they face efficiency and stability issues. Homotopy Continuation (HC) can solve more complex problems without the stability issues, and with guarantees of a global solution, but they are known to be slow. In this paper we show that HC can be parallelized on a GPU, showing significant speedups up to 56 times on polynomial benchmarks. We also show that GPU-HC can be generically applied to a range of computer vision problems, including 4-view triangulation and trifocal pose estimation with unknown focal length, which cannot be solved with elimination template but they can be efficiently solved with HC. GPU-HC opens the door to easy formulation and solution of a range of computer vision problems.*

1. Introduction

Systems of polynomial equations arise frequently in computer vision, especially in multiview geometry problems, because perspective projection is an algebraic model. Examples abound including absolute pose estimation [35, 92, 5], relative pose estimation [75, 86, 52], homography estimation [54, 14], PnP [94, 95], 3-view triangulation [17], rolling shutter camera absolute pose estimation [4], as well as many others. The challenge has been how to solve these polynomial systems efficiently and in a stable way.

The classic 5-point algorithm for relative pose estimation [78, 75] is a case in point. Its formulation begins with 15 equations in 15 unknowns, namely, 10 depths and 5 pose parameters. The traditional approach is to eliminate depths and end up with the epipolar equation which with 5 points results in a 10th-degree univariate polynomial from which pose is determined. A more formal approach to eliminating variables is the Gröbner basis [21, 22] or resultants [21, 22]. Elimination Templates were developed as an automatic solver generator [59] where the Gröbner-based elimination strategy obtained from one input is "remembered" for future inputs. These methods are reviewed in Section 2.

The challenge with the above methods is that they are limited to problems with small number of solutions. They are slow for larger problems whose elimination template can be computed. For even larger problems the computation of elimination template exceeds practical resources, rendering the problem unsolvable. In addition, stability issues might arise in the process of converting a system of polynomials to a single univariate polynomial, e.g., [71, 70].

Homotopy Continuation methods, in contrast, can solve very complex polynomial systems. The basic idea is to find all the solutions of a start system and then to continuously evolve them to the solutions of the target system. They can ensure, with probability 1, to find all solutions [84, 91], provided a "good" starting system. They also avoid the stability issues of symbolic methods as they do not manipulate the input polynomials. Their complexity depends on the number of solutions (tracks) they follow. In this lies the idea to use a GPU to speed up the computation.

GPUs have been used in computer graphics and computer vision to accelerate massively parallel operations. The key is whether HC can be parallelized to take advantage of many processors in a GPU while avoiding data transfer delays among memories. The HC process consists of prediction and correction steps in the continuation from the start system to the target system. This is done by computing the Jacobian to predict where to go next, and subsequently Newton's method to correct the solution. We show that by parallelizing the computations in the prediction and correction steps, a track can be implemented on a *warp* of a GPU. This is made possible in part by instituting kernel fusion in the MAGMA library for solving batch linear systems. In addition, expressions of homotopy Jacobians are homogenized so their evaluations can be parallelized in a single instruction multiple threads (SIMT) fashion.

Computer vision problems involving polynomial systems fit these requirements. We have applied GPU-HC to a variety of problems, and found that for moderately complex systems and beyond GPU-HC offers significant savings (with implied stability). We have also explored solutions to two problems, namely, 4-view triangulation and trifocal pose estimation with unknown focal length which have not been explored in the literature. These are introduced as example cases where elimination template fails to produce solutions but GPU-HC solves efficiently. Thus, the basic thesis of this paper is that GPU-HC can be applied to all computer vision problems formulated as polynomial systems and produce efficient and stable solutions.

2. Methods for Solving Polynomial Systems

We partition the algorithms for solving systems of polynomial equations in roughly three categories: *(i)* **Symbolic methods** that rely on algebraic elimination tools, such as Gröbner basis, resultant, *etc.*; *(ii)* **Numerical solvers** that are iterative and generally a variant of Newton's method, such as homotopy continuation, and, *(iii)* **Hybrid methods** that combine the benefits of the symbolic and numerical solvers such as elimination templates.

Symbolic solvers "transform", using algebraic elimination, the multivariate polynomial system to a univariate polynomial. The roots are then computed using dedicated algorithms like Sturm or Descartes, and are used to recover system solutions, *e.g.*, [21, 22, 82, 28]. These algorithms mainly rely on exact computations with rational numbers, performing elimination using tools such as Gröbner basis and resultants. Gröbner basis manipulate the polynomials "incrementaly" (like Gaussian elimination) to deduce the univariate polynomial, while resultants use all the polynomials right from the beginning (similar to Cramer's rule).

Symbolic methods are used widely in solving minimal problems in computer vision [49, 48, 31, 85, 40]. They always output the exact results, successfully and efficiently dealing with degeneracies such as multiple roots. However, efficient implementation of symbolic algorithms is far from a straightforward task, and systems of more than 5-6 variables of moderate degrees cannot be easily handled, except if sparsity and the structure is specifically exploited. Moreover, we are still very far from solving moderate systems in milliseconds using symbolic solvers. Another major issue, especially Gröbner basis, is that they are numerically unstable [50, 70]. This is mainly due to term-ordering that causes instability when the coefficients of the input polynomials are floating point numbers or known up to some precision. To solve such a problem, extra efforts are needed [71].

Numerical solvers are almost exclusively iterative algorithms that exploit a variant of Newton operator, performing computations in floating point arithmetic [9, 84, 91]. Approaches based on numerical linear algebra techniques, mainly on eigenvalue computations [11, 16], also belong to this family. The most prominent representative is Homotopy Continuation (HC) algorithm [7, 8, 20, 41, 91, 39]. It relies on a simple and elegant idea to initially solve a simpler polynomial system (start system) and then deform its roots to the roots of the system we want to solve (target system). Some cares are required on choosing an easy-to-solve start system with at least as many solutions as the target system. HC can handle very big problems, especially in the absence of degeneracies, say multiple roots, and is able to handle systems that are out of the reach of symbolic solvers. HC is used widely in computer vision, especially for minimal problems in multiview geometry [51, 79, 68, 26, 29, 25]. Nevertheless, HC is comparatively slow, a serious bottleneck to their wide adoption.

Numerical problems might also occur in HC algorithms, especially if the Jacobian of the system is ill-conditioned and in many cases we need to use double-precision floating point arithmetic, *e.g.*, [9]. However, HC is an inherit numerical method and does not require an exact input. Also sometimes it is not easy, if possible at all, to find good, let alone optimal, start systems, the cardinality of the output is not always correct, and extra verification steps are needed. Nevertheless, they are in general easier to be implemented than symbolic methods, even though in all the cases efficient software requires tremendous amount of time, energy, and effort to be efficient and solve real life problems.

Hybrid solvers aim to combine the symbolic and numerical approaches [27, 70, 67], and they have various algorithmic variants. A well-known method in the computer vision community is the elimination template, or automatic solver generation [55, 56, 60, 64, 59]. The main idea is to bookkeep the steps that an elimination (usually Gröbner basis) algorithm performs for one input and apply these steps to any other input. They generate a "template" of elimination at an offline stage with the random coefficients of a "dummy" system on a finite field. The solutions are then obtained by eigenvalue computations. The method is particularly fast for solving systems with low degree and low number of variables [80, 19, 93, 61, 4, 23]. Nevertheless, even though they have turned out to be quite successful in some problems, they might need to handle very large matrices [59] which are computationally intractable. Plus, it is far from trivial to analyze their stability. Several precomputations could be performed to overcome the instability issue; alas, at the end they must compute with a matrix, similar to Gröbner basis and resultants, which has a dimension at least the number of complex solutions. The condition number of such matrix is not well, if at all, studied, and it is not clear if they can handle problems with ≥ 300 roots.

3. Homotopy Continuation

The idea of Homotopy Continuation (HC) [69, 84] is to evolve the solutions of one polynomial system G, the "start system", to discover the solutions of another system F, the "target system". Let $X \in \mathbb{R}^M$ represents M unknowns. Let $F(X) = (f_1(X), f_2(X), ..., f_N(X))$ be a system of N polynomial equations, and $G(X) = (g_1(X), g_2(X), ..., g_N(X))$ be the polynomial system with known solutions. The idea of HC is to construct a series of intermediate polynomial systems $H(X,t)$ where $H(X,0) = G(X)$ and $H(X,1) = F(X)$, *e.g.*, via linear interpolation:

$$H(x,t) = (1-t)G(x) + tF(x), \qquad t \in [0,1]. \tag{1}$$

The basic idea is to find the solutions of $H(X, t+\Delta t)$ from the solutions of $H(X,t)$. Figure 1 illustrates the idea for one solution and one unknown. The black curve is the loci of the solution $X(t)$ of $H(X,t)$ forming a homotopy curve, where X_0 is the known solution of $G(X)$ and X_1 is the desired solution of $F(X)$. We track solution of $H(X,t)$ from X_0 with a number of iterations, each consisting of a prediction and a correction step. Prediction uses a first-order Taylor expansion to estimate X at $t+\Delta t$,

$$X^*(t+\Delta t) = X(t) + \tfrac{dX}{dt}\Delta t, \tag{2}$$

where X^* is the first order estimation of $X(t+\Delta t)$. We obtain $\frac{dX}{dt}$ by differentiating $H(X(t),t)$, *i.e.*,

$$\frac{\partial H}{\partial X}\frac{dX}{dt} + \frac{\partial H}{\partial t} = 0 \longrightarrow \frac{dX}{dt} = -(\frac{\partial H}{\partial X})^{-1}\frac{\partial H}{\partial t}, \tag{3}$$

where $J = \frac{\partial H}{\partial x}$ is an $M \times N$ Jacobian of H w.r.t X. This step, the first-order estimation of X^* from $X(t)$, is known as the **prediction** step (Figure 1). However, we can improve the prediction using a higher-order method like a 4-th order

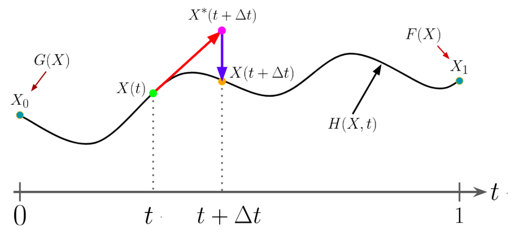


Figure 1: A track (curve) of a Homotopy Continuation algorithm showing $H(X,t)$ in black, along with one prediction (red) and one correction (blue).

Runge-Kutta. After the prediction step, $X^*(t+\delta t)$ may be far away from the homotopy curve $X(t)$. Thus, a **correction** is used to update $X^*(t+\Delta t)$ to $\hat{X}(t+\Delta t)$, *i.e.*,

$$H(X^*, t+\Delta t) + \tfrac{\partial H}{\partial X}(X^*, t+\Delta t)(\hat{X} - X^*) = 0, \tag{4}$$

using Newton's method, giving $\hat{X}$ in the form of

$$\hat{X} = X^* - (\tfrac{\partial H}{\partial X})^{-1}(X^*, t+\Delta t)H(X^*, t+\Delta t). \tag{5}$$

The pairs of prediction and correction steps numerically evolve X_0 as the solution of $G(X)$ to X_1 as the solution of $F(X)$ with t goes from 0 to 1 iteratively.

Provided that we have a good started system, the HC algorithm finds all the solutions (up to some approximation) with probability one. More detailed descriptions of HC algorithm can be found in [13].

4. Illustrative Problems

Preliminaries: Let Γ denote a 3D point which projects to an image point $\gamma = (\xi, \eta, 1)^T$ with depth ρ so that $\Gamma = \rho\gamma$. The expression of Γ in a camera related by pose (R,T) to another camera where R is the rotation matrix and T is translation, is $\bar{\Gamma} = R\Gamma + T$. Due to metric ambiguity the unit direction $\hat{T}$ along T in sought, where $T = \lambda\hat{T}$.
Relative Pose Estimation with Calibrated Cameras is a classic problem most frequently solved by Nister's 5-point algorithm [74, 75, 78]. Consider five corresponding points $(\gamma_i, \bar{\gamma}_i)$ where γ_i in one image is in correspondence with $\bar{\gamma}_i$ in another image. Since $\Gamma_i = \rho_i\gamma_i$, $\bar{\Gamma}_i = \bar{\rho}_i\bar{\gamma}_i$, and $\bar{\Gamma}_i = R\Gamma_i + T$. The relationship between γ_i and $\bar{\gamma}_i$ is captured as

$$\bar{\rho}_i\bar{\gamma}_i = R\rho_i\gamma_i + \hat{T}, \qquad i = 1, 2, \cdots, N, \tag{6}$$

where the depths (ρ_i,$\bar{\rho}_i$) represent 10 unknowns and (R,$\hat{T}$) represent 5 unknowns. The above set of five vector equations give 15 constraints in 15 unknowns. Representing R with quaternions which involves 4 unknowns with one equation yields 16 *polynomial equations* in 16 unknowns. Observe that there has been no attempts in the literature to solve these equations, which HC can solve, referred to as "5 pt rel. pose & depth recon." in Table 2. Rather, the traditional approach is to reduce the number of unknowns by eliminating the ten depth variables by taking cross product of Equation 6 with $\hat{T}$ and then dot product with $\bar{\gamma}_i$ giving the classical epipolar relationship, *i.e.*, $\bar{\gamma}_i^T E \gamma_i = 0$, $i = 1, 2, \cdots, 5$, where $E = [\hat{T}]_\times R$. While this is 5 equations in 5 unknowns , these involve trigonometric equation unless R is represented by a quaternion giving 6 polynomial equations in 6 unknowns. Again, this can also be solved by HC. Still, the classic approach is to treat E as nine unknowns and use $E = [\hat{T}]_\times R$ from [74] if and only if

$$2EE^TE - trace(EE^T)E = 0. \tag{7}$$

These are 9 cubic polynomial equations but only four are independent which can be used in conjunction with the classical epipolar relationship to solve for E. Namely, E is written in a vector form as $\tilde{E}$,

$$\begin{cases} \tilde{E}^T = [E_{11}, E_{12}, E_{13}, E_{21}, E_{22}, E_{23}, E_{31}, E_{32}, E_{33}] \\ w_i^T \tilde{E} = 0, \qquad i = 1, 2, \cdots, 5, \\ w^T = [\xi_i \bar{\xi}_i, \eta_i \bar{\xi}_i, \bar{\xi}_i, \xi_i \bar{\eta}_i, \eta_i \bar{\eta}_i, \bar{\eta}_i, \xi_i, \eta_i, 1]. \end{cases} \tag{8}$$

$\tilde{E}$ is then an arbitrary linear sum of the four matrices representing the right nullspace, $\tilde{E} = \alpha_1 E_1 + \alpha_2 E_2 + \alpha_3 E_3 + E_4$, where the last constant α_4 is set to one due to the scale invariance of E. The only remaining constraint is the set of nine cubic Equations 7, where the unknowns (α_1, α_2, α_3) involve 20 monomials up to order 3, so that they can be expressed as a 9×20 matrix multiplied by a vector of 20 monomials. Then, all monomials can be eliminated except those involving one variable, say α_3. This can be done by Gauss-Jordan elimination with partial pivoting to make an upper triangular matrix which can lead to a single tenth-order polynomial in one variable α_3 by manually derived Gröbner basis, giving 10 roots. The real root of α_3 can solve α_1, α_2 and E from which R and T can be recovered.

Li and Hartley [65] solve Equation 7 with $\tilde{E}$ as described by Equation 8 using the hidden variable technique, a resultant technique for algebraic elimination [21]. They include $det(E) = 0$ as a tenth equation and solve equating the determinant of the 10×10 matrix to zero as a function of α_3, a tenth-order polynomial which can then be solved. The claimed advantage of this technique over Nister's is its simplicity and ease of implementation.

Observe that both approaches devise ingenius algorithms to turn the basic system of polynomial Eqaution 6 into a single 10th degree uni-variate polynomial. In contrast, Homotopy Continuation can be used immediately to solve 16×16 polynomial system or the reduced 6×6 system of Equation 7. Finally, HC can also be used to solve (α_1, α_2, α_3) using a 3×3 system of cubic polynomials. Note that we are not advocating to solve the relative pose using HC (the system is too small to benefit from it). Rather, we are noting that it *can* be solved by HC as an illustration.

Perspective-n-Point problem (PnP) estimates the pose of a calibrated camera (R,T) using n correspondences between 3D world coordinate points Γ_i and their 2D projections in the image γ_i. The P3P problem where 3D points (Γ_i, Γ_2, Γ_3) correspond to 2D image points (γ_1, γ_2, γ_3), respectively, has a long history [33, 32, 36, 81] and it has 4 solutions requiring a 4th correspondence to disambiguate.

The basic formulation can be posed using $\Gamma_i = \rho_i \gamma_i$, where $i = 1, 2, 3$. This is a set of nine equations in nine unknowns. At this point, HC can be used to solve for (R,T) and depths. Using a quaternion representation of R which involves 4 unknowns and one equation, this becomes a set of 10×10 polynomials with 10 unknowns. The traditional approach eliminates R and T to solve depths from

$$\begin{cases} (\Gamma_2 - \Gamma_1)^T(\Gamma_2 - \Gamma_1) = (\rho_2\gamma_2 - \rho_1\gamma_1)^T(\rho_2\gamma_2 - \rho_1\gamma_1) \\ (\Gamma_3 - \Gamma_1)^T(\Gamma_3 - \Gamma_1) = (\rho_3\gamma_3 - \rho_1\gamma_1)^T(\rho_3\gamma_3 - \rho_1\gamma_1) \\ (\Gamma_2 - \Gamma_1)^T(\Gamma_3 - \Gamma_1) = (\rho_2\gamma_2 - \rho_1\gamma_1)^T(\rho_3\gamma_3 - \rho_1\gamma_1) \end{cases}, \tag{9}$$

a set of three quadratics in three unknowns (ρ_1,ρ_2,ρ_3). Again, this reduced form can be easily solved by HC, but the traditional approach is to apply Silvester's resultant to get an 8th degree polynomial, containing even terms so that it is effectively a quartic [81].

The general PnP problem relies on n correspondences between 3D points Γ_i and 2D image points γ_i, $i = 1, 2, ..., n$. A direct minimization of the algebraic reconstruction error [94] uses a non-unit quaternion representing of R and explicitly optimize for R. This gives four polynomials of degree three in four variables, which are solved by Gröbner bases, from which an elimination template is constructed using the automatic generator in [55]. This gives at most 81 solutions with an 575×656 elimination template and 81×81 action matrix. Alternatively, these equations can be solved using HC without any further processing with around a factor of 5 times speedup on a GPU, Table 2. In this larger case, HC features both simplicity and efficiency.

N-view Triangulation aims to find the 3D world point Γ that is most consistent with a set of projection, $\gamma_1, \cdots, \gamma_N$ from N views, given relative pose of all cameras in the form of the pairwise essential matrix E_{ij} between views i and j. Due to noise, the projection rays from corresponding points do not necessarily meet in space. For two views, using mid-point between the closest points on the projection rays [10] could be erroneous, especially with large calibration error. Rather than minimizing the latent 3D error, reprojection error can be minimized [37, 38, 46]. Let $\gamma_i = \hat{\gamma}_i + \Delta\gamma_i$ where $\hat{\gamma}_i$ is the true 2D observation and $\Delta\gamma_i$ is the error introduced by noise, *i.e.*,

$$\hat{\gamma}_j^T E_{ij} \hat{\gamma}_i = 0, \; (\gamma_j - \Delta\gamma_j)^T E_{ij} (\gamma_i - \Delta\gamma_i) = 0. \tag{10}$$

Minimizing reprojection errors $\Delta\gamma_i$ and $\Delta\gamma_j$ solves

$$(\Delta\gamma_i^*, \Delta\gamma_j^*) = \underset{(\gamma_j - \Delta\gamma_j)^T E_{ij} (\gamma_i - \Delta\gamma_i) = 0}{\arg\min} (||\Delta\gamma_i||^2 + ||\Delta\gamma_j||^2).$$

Using Lagrange multipliers and notation $\Delta\gamma_i^T = (u_i, v_i, 0)$ the problem becomes

$$(u_i^*, v_i^*, u_j^*, v_j^*, \lambda^*) = \underset{u_i, v_i, u_j, v_j, \lambda^*}{\arg\min} \; (u_i^2 + v_i^2 + u_j^2 + v_j^2)$$
$$+ \lambda(\gamma_j^T - [u_j, v_j, 0])E_{ij}(\gamma_i - \begin{bmatrix} u_i & v_i & 0 \end{bmatrix}^T).$$

Setting the first derivative w.r.t the five variables gives a 5×5 polynomial system. This system can be solved using HC with ease effort. Traditionally, however, the system is solved by eliminating four of five variables, gives a single 6-th order polynomial [38]. This gives excellent results but it is slow prompting [46, 66] to use an iterative method which is faster but is prone to being stuck in local minima.

The N-view triangulation is not as well-explored despite the formulation of minimizing reprojection error is identical

$$(\Delta\gamma_1^*, \Delta\gamma_2^*, \cdots, \Delta\gamma_N^*) = \tag{11}$$
$$\underset{\substack{\Delta\gamma_1, \Delta\gamma_2, ..., \Delta\gamma_N \\ such\ that\ \forall i,j (\gamma_j - \Delta\gamma_j)^T E_{ij}(\gamma_i - \Delta\gamma_i) = 0}}{\arg\min} \sum_{k=1}^{N} |\Delta\gamma_k|^2,$$

$$\text{or } (u_1^*, v_1^*, u_2^*, v_2^*, \cdots, u_N^*, v_N^*) = \tag{12}$$
$$\underset{u_1^*, v_1^*, u_2^*, v_2^*, \cdots, u_N^*, v_N^*, \lambda_k}{\arg\min} \sum_{k=1}^{N} [(u_k^2 + v_k^2) +$$
$$\sum_{i=1}^{N} \sum_{j=i+1}^{N} \lambda_k(\gamma_j^T - [u_j, v_j, 0])E_{ij}(\gamma_i - \begin{bmatrix} u_i \\ v_i \\ 0 \end{bmatrix}).$$

Note that there are $2N + \frac{N(N-1)}{2} = \frac{N^2+3N}{2}$ unknowns and setting first derivatives to zero gives 5×5, 9×9 and 14×14, for two, three, and four views, respectively, becoming exponentially more difficult to use with Gröbner basis and other traditional methods. [58] restricts the consideration of all sequential pairwise essential matrices to these with the previous view, *i.e.*, E_{12}, E_{23}, *etc.* and uses the elimination template method with a 274×305 template; [59] reduces the size of the elimination template to 239×290. Note that the full problem gives an elimination template of 1866×1975 which is impractical to solve. Similarly, the 4-view triangulation gives improved error but it leads to large polynomial systems. Homotopy Continuation, however, can solve these problems and with improved efficiency, Table 2.

Trifocal relative pose estimation with unknown focal length aims to estimate the relative poses between three views as well as the focal length. Trifocal pose estimation has drawn attentions recently [63, 18, 45, 47, 30]. These approaches often assume that the intrinsic matrix is available. Recently, [62] estimates trifocal tensor with radial distortion, a minimal problem of one pinhole camera and two radial cameras. We consider another minimal problem with three pinhole cameras and one common focal length, which needs only 4 points correspondences across three views. Let the calibration matrix be $K = diag(f, f, 1)$, where f is the focal length. Consider three corresponding points (γ_1, γ_2, γ_3) in image (1,2,3), respectively, with unknown depth (ρ_1, ρ_2, ρ_3) respectively. Then, denoting (R_{12}, T_{12}) and (R_{13}, T_{13}) the relative pose of the second and third cameras w.r.t the first, respectively, we have

$$\begin{cases} \rho_2 K^{-1}\gamma_2 = \rho_1 K^{-1} R_{12}\gamma_1 + \hat{T}_{12} \\ \rho_3 K^{-1}\gamma_3 = \rho_1 K^{-1} R_{13}\gamma_1 + T_{13}, \end{cases} \tag{13}$$

where $\hat{T}_{12}$ is taken to have unit length. Thus, there are 11 poses and 3 depth unknowns. Since there are four sets of correspondences, there is a total of 24 unknowns including one f, 11 from poses and 12 depths. There are also four sets of vector, Equation 13, which each gives 6 equations. If R_{12} and R_{13} are represented by quaternions, we have 26 equations in 26 unknowns which can be solve by HC.

Alternatively, ρ_2 and ρ_3 can be written in terms of ρ_1 as

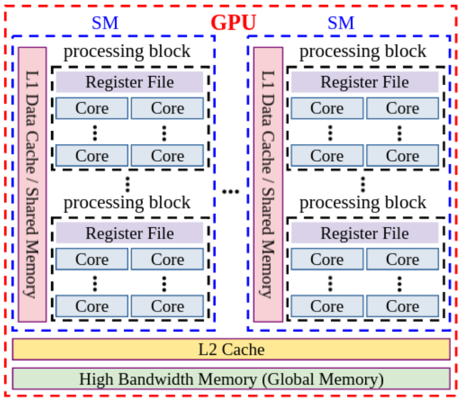
$$\begin{cases} \rho_2 e_3^T K^{-1}\gamma_2 = e_3^T \rho_1 K^{-1} R_{12}\gamma_1 + e_3^T K^{-1} T_{12} \\ \rho_3 e_3^T K^{-1}\gamma_3 = e_3^T \rho_1 K^{-1} R_{13}\gamma_1 + e_3^T K^{-1} T_{13}. \end{cases} \tag{14}$$

Substituting these back into Equation 13 gives 4 equations for each triplet of correspondings for a total of 16 equations. The unknowns are 1 focal length, 11 pose and 4 depths for a total of 16 unknowns. If R is represented as a quaternion, one additional unknown and one additional equations are introduced per rotation matrix, giving a total of 18 polynomial equations in 18 unknowns. This minimal problem cannot be solved by elimination template since it requires out of bounds memory even on a high performance computing machine. However, our HC implementation can solve this system with 3326 ms in CPU and 388 ms in GPU, Table 2.

5. GPUs and Computer Vision

GPUs are often preferred over CPUs because of their superior computational power, memory bandwidth, and energy efficiency. For example, a V100 GPU provides an FP64 compute peak of 7 TFlop/s and memory bandwidth of 900 GB/s at 250 Watts. While one CPU core is faster and provides wider instruction sets, GPUs have many more cores, *e.g.*, 5,120 in the V100. The key to unlocking the computational power of the GPU is to design algorithms that are highly parallel and use efficiently all the cores.

Figure 2 shows the GPU architecture. The CUDA cores are organized into Streaming Multiprocessors (SMs) where each SM has a number of CUDA cores. The GPU work is organized into *kernels* that have two levels of nested parallelism - a coarse level that is data parallel and is spread across the SMs, and a fine level within each SM. The parallelism is organized in terms of thread blocks (TBs). A TB is scheduled for execution on one of the SMs and is data parallel with respect to the other TBs. Each TB is composed of multiple threads running in groups of 32 called *warps*.


GPU
SM
SM
processing block
Register File
Core
L1 Data Cache / Shared Memory
L2 Cache
High Bandwidth Memory (Global Memory)
Figure 2: The NVIDIA's GPU architecture and memory hierarchy. In Tesla V100 GPU, there are 80 SMs each partitioned into four processing blocks, each having 16 cores. The register files per SM, shared memory/L1 cache per SM, L2 cache, and global memory are respectively 256KB and 96KB, 6,144KB and 16GB, with hit latencies of 3 ns, 22.68 ns, 156.33 ns, and roughly 366 ns, respectively [44].

Threads in a TB can share data through a shared memory module. Private variables that have the scope of one thread are usually stored in the register file. Algorithms must be designed to support this type of parallelism.

The multi-level memory hierarchy enables compute-bound operations, like general matrix-matrix multiplication (GEMM), perform close to the compute peak of the GPU [73]. Subsequently, many dense linear algebra algorithms such as the ones in LAPACK, and subsequently MAGMA (LAPACK for GPUs), can be expressed using GEMMs and BLAS in general [89]. Some numerical algorithms, like the one mentioned in Section 6, involve many independent computations (*e.g.* dense factorizations) on relatively small matrices. These algorithms, are limited by the memory bandwidth, but have a high degree of parallelism, which is suitable for GPUs. Maximizing data reuse is possible by caching each matrix entirely in the register file or shared memory, which enables GPUs to outperform multicore CPUs in these types of algorithms.

Algorithms in computer vision are naturally data-parallel and computationally intensive, and therefore a good fit for modern GPUs. Computational patterns involving one-to-one mappings like an image convolved by different filters can benefit from the data parallelism and the memory hierarchy. Many-to-one mappings that involve summations of certain buffers also can benefit the memory hierarchy and do it fast in multi-processors. Many-to-many computational patterns can be mapped efficiently to GPUs as well [77]. Operations that can not be mapped efficiently to GPU have been left in general for the CPUs. This usually involves irregular computations on small data sets with insufficient parallelism, and computations with heavy data dependencies (like solving a small system of equations). Still, techniques like batching computations to increase parallelism and developments in numerical linear algebra libraries for GPUs [1, 34], have laid the groundwork for many more algorithms to be easily ported and benefit GPU use. Often algorithms that have been avoided before due to their computational cost are becoming preferred for GPUs when current advances make their GPU mapping very efficient. This is the case for the HC that we target to develop.

6. GPU Implementation of HC

Homotopy continuation process can be parallelized in two ways: First, observe that since HC follows many independent tracks to convergence, a straightforward approach would be to assign each track to a thread. However, the efficiency of GPU processing depends on *(i)* number of threads processing many tracks in parallel, and *(ii)* avoiding costly data transfer rates by using the fast register files, or at least L1 caches v.s. the slower L2 cache or even slower global memory, Figure 2. In our application, each track requires a few Kbytes while the available memory is 125, 46, 37, and 97K bytes for register file, L1 cache, L2 cache, and global memory, respectively, for one thread per track. Thus any process requiring more than 125 + 46 + 37 = 208 bytes of memory is forward to use the very slow global memory. As a result, each track must make use of many threads, and not only the processing must be parallelized, but so must the use of memory with the aim of keeping everything in register file, shared memory, or at least L1 cache.

Observe that the other extreme of spreading a trade over numerously many threads starts becoming counterproductive because the synchronization of threads employs the slower shared memory (2 clock cycles per 32 threads) so that if 2048 threads are employed per track, 128 clock cycles ($\sim$104 ns) times the tens of thousands of their synchronization is needed which becomes an unnecessary overhead.

The optimal balance for the target applications is to assign a track to a warp (32 threads) using one GPU core. This gives the application access to $256K/64 = 4K$ very fast register file memory and $96K/64 = 1.5K$ of fast L1 cache (if no shared memory is used), well satisfying the memory requirement of the target application. On the other hand, the cost of thread synchronization is only 2 clock cycles.

The second intuition aims for parallelizing HC within each warp by *(i)* solving a system of linear equations in both the prediction step, Eq. 3 and the correction step, Eq. 4, and *(ii)* evaluating the Jacobian matrix $\partial H/\partial x$, $\partial H/\partial t$, and the homotopy H, Eq. 2 and 5.

Linear System Solver: The vast majority of work on solving linear systems on GPU is centered around large matrices, motivating a hybrid CPU+GPU approach [88, 90]. For smaller matrices like ours, cuBLAS or MAGMA [3, 2, 43] can be used. A linear system is generally solved by an LU factorization with partial pivoting followed by two triangular solves. The LU factorization in MAGMA is fast, typically 15% to 80% faster than cuBLAS for small matrices. However, we found out that cuBLAS is faster than MAGMA for the combined (factorization + solve) operation. This is mainly due to a slow triangular solver kernel in MAGMA, which does not take advantage of small matrices.

Our contribution to improving these standard libraries for our purposes is twofold. First, solving the linear system as two separate GPU kernels causes redundant global memory traffic. The two kernels can be fused into one if the matrices are small, thus maximizing data reuse in the register file. The proposed *kernel fusion* significantly speeds up the solution. Second, in solving a linear system $Ax = b$, the decomposition can act on the augmented matrix $[A\ b]$, which implicitly carries out the triangular solve with respect to the L factor of A. The second triangular solve uses the cached U factor after the factorization is complete. The proposed fused kernel is now integrated into the MAGMA library.

Parallel Evaluations of the Jacobian and Vectors: The main bottleneck to parallel evaluations of the elements of the Jacobian matrix $\partial H/\partial x$ and the vectors $\partial H/\partial t$ and H is the heterogenuity of its elements which prevents evaluation by many threads requiring a uniform format. This heterogeneity can be illustrated by a simple example of a system with two variables $X = (x_1, x_2)$ where the Jacobian elements are spanned by monomials, for example, A = $a_1x_1 + a_2x_1x_2 + a_3x_2^2$ or B = $a_4x_1x_2 + a_5x_2^2$, where the coefficients a_i are linear interpolation of corresponding elements in the start and target systems. A straightforward approach to homogenize these expressions is to write each as a sum over all possible monomials and associate a scalar zero with those absent from the Jacobian elements. However, due to the extreme sparsity, the process is inefficient.

Alternatively, consider K the maximum number of terms in the Jacobian matrix elements; in the above examples, A has three terms and B has two terms, so that $K = 3$ if these were the only elements of the Jacobian matrix. Furthermore, consider that each term consists of a scalar multiplied with a coefficient and a number of variables, *e.g.*, the third term of A is a product of $(1, a_3, x_3, x_3)$ while the first term of B is $(1, a_4, x_1, x_2)$. Note that the first term of A is a product of $(1, a_1, x_1)$. Thus, to homogenize the expression, it is written as $(1, a_1, x_1, x_3)$ where the auxiliary variable $x_3 = 1$. Now all terms of both A and B can be written as

$$U = \sum_{k=1}^{K} s_k a_{k,j} x_{k,m_1} x_{k,m_2} \cdots x_{k,m_M},$$

where s_k is a scalar, $a_{k,j}$ identifies a coefficient, x_{k,m_i} identifies one of the variables, including $x_3 = 1$, and M is the maximal number of variables in a term. With this in mind the only data to be communicated for the parallel computation of U is $(s_k, a_{k,j}, x_{k,m_1}, x_{k,m_2}, ..., x_{k,m_M})$ where $a_{k,j}$, x_{k,m_i} are pointers to data stored in shared memory and accessed by an index, *i.e.*, A is represented by $((1,1,1,3), (1,2,1,2), (1,3,2,2))$ and B is represented by $((1,4,1,2), (1,5,2,2), (0,1,1,1))$. Note that $\partial H/\partial t$ and H are evaluated in the same way although the coefficients a_k are different. This homogeneous form allows for parallel computation of all elements of the Jacobian matrix $\partial H/\partial x$ and the vectors $\partial H/\partial t$ and H.

Finally, there is an issue on how to allocate the parallel computations per thread. Recall that each track is assigned to a warp which has 32 threads. Since the matrices are generally less than 32×32, and since the subsequent operation of LU decomposition is row-by-row with one thread per row, it makes sense to assign one row per thread.

7. Experiments

The experiments aim at testing kernel fusion for batch linear systems, and measuring performances on polynomial system benchmarks as well as computer vision problems. We use an 8-core 2.6GHz Intel Xeon CPU and an nVidia Quardro RTX 6000 GPU, unless otherwise specified.

Kernel-Fused Batch Linear Systems: The performance of the batched linear systems with kernel fusion and augmented matrix, Section 6, is compared with cuBLAS and MAGMA in Figure 3 on a Tesla V100-PCIe GPU for 1000 matrices with sizes ranging from 4×4 to 20×20. Evidently, kernel-fused MAGMA outperforms cuBLAS with speedup of $2.23\times$ to $3.65\times$ and MAGMA with speedups ranging from $3.11\times$ to $4.91\times$.

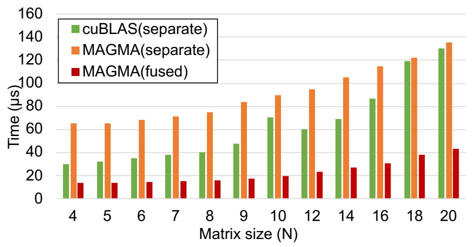


Figure 3: The performances of the batch linear systems of MAGMA with kernel fusion, MAGMA, and cuBLAS.

Problems	# of Unkns.	# of Sols.	CPU (ms)	GPU (ms)	$\frac{\text{CPU}}{\text{GPU}}$
alea6 [76]	6	387	105.67	2.02	52.31×
cyclic7 [12]	7	924	177.95	4.77	37.31×
katsura10 [42]	11	1024	414.12	7.34	56.42×
eco12 [69]	12	1024	227.54	17.06	13.34×

Table 1: Performance of GPU-HC on benchmark problems.

Polynomial System Benchmarks: We selected four representative benchmark polynomial systems [76, 12, 42, 69] to evaluate our GPU-HC. Table 1 shows GPU-HC significant speedup ranging from $13\times$ to $56\times$. Figure 4 (a) shows the residual of evaluating each polynomial system averaged

Problems	# of Unkns.	# of Sols.	Elim. Temp. (ms)	CPU (ms)	GPU (ms)	$\frac{\text{CPU}}{\text{GPU}}$	$\frac{\text{Elim. Temp.}}{\text{GPU}}$
trifocal rel. pose, unknown focal length	18	1784	X	3326.07	**338.27**	10.21×	N./A.
4-view triangulation	14	296	X	156.28	**18.60**	8.32×	N./A.
5 pt rel. pose & depth recon.	16	160	X	150.94	**26.89**	5.61×	N./A.
6 pt rolling shutter abs. pose w. 1-lin. [6]	18	160	X	158.48	**27.11**	5.85×	N./A.
3-view triangulation [17]	9	94	612.432	101.86	**8.17**	24.19×	38.24 ×
optimal PnP with quaternion [72]	4	128	36.329	80.26	**7.18**	11.18×	5.06×
P4P, unknown focal length & radial distortion [15]	5	192	9.03	130.79	**7.51**	17.42×	1.2×
2-view triangulation with radial distortion [57]	5	28	5.92	66.22	**3.06**	21.64×	1.93×
optimal P4P abs. pose [87]	5	32	1.864	53.13	**1.57**	33.84×	1.19×
3 pt rel. pose w. homography constraint [83]	8	8	1.472	51.25	**0.95**	53.95×	1.55×
rel. pose w. quiver, unknown focal length [53]	4	28	**1.082**	56.01	1.23	45.54×	0.88×
P3P abs. pose [55]	3	8	**0.063**	39.64	0.22	180.18×	0.29×
5 pt rel. pose w.o. depth recon. [75]	3	27	**0.035**	55.48	0.96	57.79×	0.036×

X: it is impossible for elimination template to solve because of an out of memory issue

Table 2: Performance of GPU-HC and CPU-HC *vs.* elimination template in application to several minimal problems.

from 100 times with random inputs for both start and target systems. It reveals that the speedup is not at the cost of lower accuracy, *i.e.*, the GPU-HC computed solutions satisfy the polynomial system with high accuracy.

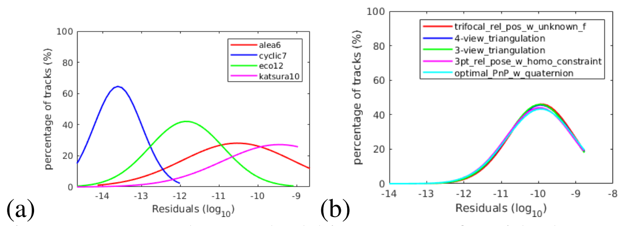


Figure 4: Normal smoothed histograms of residual errors over all the solutions solved by GPU-HC. (a) Polynomial system benchmarks (b) Computer vision problems.

Computer Vision Problems: We consider a sample of minimal problem in computer vision ranging from the classic pose estimation P3P to the more complex 3-view triangulation as well as two problems that have not been explored previously: *(i)* 4-view triangulation is an extension of 3-view triangulation [17]. As far as we know, this is the first attempt to explore this problem. *(ii)* trifocal relative pose estimation with unknown focal length is an extension of existing trifocal relative pose estimation problems [29, 62] to the uncalibrated scenario; as far as we know, this problem has not been explored previously. These two problems were introduced in Section 4. The most popular technique for solving polynomial system is the elimination template approach [59] and is used to gauge the performances of GPU-HC. Table 2 shows a comparison of the elimination template performances with that of HC on CPU and on GPU. Each problem is instantiated 20 times with random parameters and its performance is averaged. The start systems for HC are generated with monodromy module in Macaulay2 [24]. Numerous factors affect the speedup of GPU-HC over CPU-HC, including the number of solutions, number of unknowns, and the number of terms in the polynomial evaluations of the Jacobian matrix. The performances of elimination template is dependent on the size of the linear system it solves which itself is related to the number of solutions of the polynomial system. Note that for the top four problems the elimination template cannot compute the basis of the quotient ring of the system even with ample memory. A review of Table 2 which is ordered by elimination template time, shows that with the exception simpler problem such as P3P (the bottom four rows), the GPU-HC outperforms the elimination template. GPU-HC opens the door to more complex problems that the elimination template cannot handle. Furthermore, Figure 4 (b) is the resulting residual histograms on selected problems, showing that the speedup of GPU-HC is also not at the cost of giving promising estimations. Additional experimental data can be found in the supplementary materials.

8. Conclusion

We presented GPU-HC, a GPU implementation of HC that is generic and can be easily applied to any computer vision problem formulated as a system of polynomial equations. The significant speedup of GPU-HC is an enabler in that HC can now be efficiently used for moderately complex problems in place of completing approaches. GPU-HC also enables the exploration of problems whose complexity has thus far evaded a practical solution.

Acknowledgement
Kimia, Fan, and Chien gratefully acknowledge the support of NSF award 1910530. Tsigaridas is partially supported by ANR JCJC GALOP (ANR-17-CE40-0009). Abdelfattah and Tomov are partially supported by NSF Grant No. OAC 1740250.

References

[1] A. Abdelfattah, A. Haidar, S. Tomov, and J. Dongarra. Performance, Design, and Autotuning of Batched GEMM for GPUs. In *High Performance Computing - 31st International Conference, ISC High Performance 2016, Frankfurt, Germany, June 19-23, 2016, Proceedings*, pages 21–38, 2016. 6

[2] E. Agullo, J. Demmel, J. Dongarra, B. Hadri, J. Kurzak, J. Langou, H. Ltaief, P. Luszczek, and S. Tomov. Numerical linear algebra on emerging architectures: The PLASMA and MAGMA projects. *J. Phys.: Conf. Ser.*, 180(1), 2009. 6

[3] A. Ahmad, H. Azzam, T. Stanimire, and D. Jack. Batched one-sided factorizations of tiny matrices using gpus: Challenges and countermeasures. *Journal of Computational Science*, 26:226–236, 2018. 6

[4] C. Albl, Z. Kukelova, V. Larsson, and T. Pajdla. Rolling shutter camera absolute pose. *IEEE transactions on pattern analysis and machine intelligence*, 42(6):1439–1452, 2019. 1, 3

[5] C. Albl, Z. Kukelova, and T. Pajdla. R6p-rolling shutter absolute camera pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2292–2300, 2015. 1

[6] V. L. Albl Cenek, Kukelova Zuzana and T. Pajdla. Rolling shutter camera absolute pose. *IEEE transactions on pattern analysis and machine intelligence*, 42(6):1439–1452, 2019. 8

[7] J. Alexander and J. A. Yorke. The homotopy continuation method: numerically implementable topological procedures. *Transactions of the American Mathematical Society*, 242:271–284, 1978. 2

[8] D. J. Bates, J. D. Hauenstein, A. J. Sommese, and C. W. Wampler. Adaptive multiprecision path tracking. *SIAM Journal on Numerical Analysis*, 46(2):722–746, 2008. 2

[9] D. J. Bates, A. J. Sommese, J. D. Hauenstein, and C. W. Wampler. *Numerically solving polynomial systems with Bertini*. SIAM, 2013. 2

[10] P. A. Beardsley, A. Zisserman, and D. W. Murray. Navigation using affine structure from motion. In *European Conference on Computer Vision*, pages 85–96. Springer, 1994. 4

[11] M. R. Bender and S. Telen. Yet another eigenvalue algorithm for solving polynomial systems. *arXiv preprint arXiv:2105.08472*, 2021. 2

[12] G. Björck and R. Fröberg. A faster way to count the solutions of inhomogeneous systems of algebraic equations, with applications to cyclic n-roots. *Journal of Symbolic Computation*, 12(3):329–336, 1991. 7

[13] P. Breiding and S. Timme. Homotopycontinuation. jl: A package for homotopy continuation in julia. In *International Congress on Mathematical Software*, pages 458–465. Springer, 2018. 3

[14] M. Brown, R. I. Hartley, and D. Nistér. Minimal solutions for panoramic stitching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 1

[15] M. Bujnak, Z. Kukelova, and T. Pajdla. New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion. In *Asian Conference on Computer Vision*, pages 11–24. Springer, 2010. 8

[16] L. Busé, H. Khalil, and B. Mourrain. Resultant-based methods for plane curves intersection problems. In *International Workshop on Computer Algebra in Scientific Computing*, pages 75–92. Springer, 2005. 2

[17] M. Byröd, K. Josephson, and K. Åström. Fast optimal three view triangulation. In *Asian conference on computer vision*, pages 549–559. Springer, 2007. 1, 8

[18] J. Chen, B. Jia, and K. Zhang. Trifocal tensor-based adaptive visual trajectory tracking control of mobile robots. *IEEE transactions on cybernetics*, 47(11):3784–3798, 2016. 5

[19] L. Chen, Y. Zheng, A. Subpa-Asa, and I. Sato. Polarimetric three-view geometry. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–36, 2018. 3

[20] T. Chen, T.-L. Lee, and T.-Y. Li. Hom4ps-3: a parallel numerical solver for systems of polynomial equations based on polyhedral homotopy continuation methods. In *International Congress on Mathematical Software*, pages 183–190. Springer, 2014. 2

[21] D. Cox, J. Little, and D. OShea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer Science & Business Media, 2013. 1, 2, 4

[22] D. A. Cox, J. B. Little, and D. O'Shea. *Using algebraic geometry*. Number 185 in Graduate texts in mathematics. Springer, New York, 2nd ed edition, 2005. 1, 2

[23] Y. Ding, J. Yang, and H. Kong. An efficient solution to the relative pose estimation with a common direction. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11053–11059. IEEE, 2020. 3

[24] T. Duff. *Applications of monodromy in solving polynomial systems*. PhD thesis, Georgia Institute of Technology, 2021. 8

[25] T. Duff, K. Kohn, A. Leykin, and T. Pajdla. Plmp-point-line minimal problems in complete multi-view visibility. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1675–1684, 2019. 2

[26] T. Duff, K. Kohn, A. Leykin, and T. Pajdla. PL1P-Point-Line minimal problems under partial visibility in three views. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVI 16*, pages 175–192. Springer, 2020. 2

[27] M. Elkadi and B. Mourrain. Symbolic-numeric methods for solving polynomial equations and applications. In *Solving Polynomial Equations*, pages 125–168. Springer, 2005. 2

[28] M. Elkadi and B. Mourrain. *Introduction à la résolution des systèmes polynomiaux*, volume 59 of *Mathématiques et Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. 2

[29] R. Fabbri, T. Duff, H. Fan, M. H. Regan, D. d. C. d. Pinho, E. Tsigaridas, C. W. Wampler, J. D. Hauenstein, P. J. Giblin, B. Kimia, et al. TRPLP-trifocal relative pose from lines at points. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12073–12083, 2020. 2, 8

[30] R. Fabbri, T. Duff, H. Fan, M. H. Regan, D. d. C. d. Pinho, E. Tsigaridas, C. W. Wampler, J. D. Hauenstein, P. J. Giblin, B. Kimia, et al. TRPLP-trifocal relative pose from lines at points. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12073–12083, 2020. 5

[31] R. Fabbri, P. Giblin, and B. Kimia. Camera pose estimation using first-order curve differential geometry. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 2

[32] S. Finsterwalder and W. Scheufele. Das rückwärtseinschneiden im raum. verlag d. *Bayer. Akad. d. Wiss*, 1903. 4

[33] J. A. Grunert. Das pothenotische problem in erweiterter gestalt nebst bber seine anwendungen in der geodasie. *Grunerts Archiv fur Mathematik und Physik*, pages 238–248, 1841. 4

[34] A. Haidar, P. Luszczek, S. Tomov, and J. Dongarra. Towards batched linear solvers on accelerated hardware platforms. In *Proceedings of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP 2015, San Francisco, CA, 02/2015 2015. ACM, ACM. 6

[35] S. Haner and K. Astrom. Absolute pose for cameras under flat refractive interfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1428–1436, 2015. 1

[36] R. M. Haralick, D. Lee, K. Ottenburg, and M. Nolle. Analysis and solutions of the three point perspective pose estimation problem. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 592–593. IEEE Computer Society, 1991. 4

[37] R. I. Hartley and P. Sturm. Triangulation. In *International Conference on Computer Analysis of Images and Patterns*, pages 190–197. Springer, 1995. 4

[38] R. I. Hartley and P. Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997. 4, 5

[39] J. D. Hauenstein and M. H. Regan. Adaptive strategies for solving parameterized systems using homotopy continuation. *Applied Mathematics and Computation*, 332:19–34, 2018. 2

[40] M. HenrikStewénius, K. Aström, and D. Nistér. Solutions to minimal generalized relative pose problems, 2005. 2

[41] R. J. Holt, A. N. Netravali, and T. S. Huang. Experience in using homotopy methods to solve motion estimation problems. In *Curves and Surfaces in Computer Vision and Graphics*, volume 1251, pages 219–226. International Society for Optics and Photonics, 1990. 2

[42] H. Hong and V. Stahl. Safe starting regions by fixed points and tightening. *Computing*, 53(3):323–335, 1994. 7

[43] MAGMA: Matrix Algebra on GPU and Multicore Architectures. Available at http://icl.cs.utk.edu/magma/. 6

[44] M. M. B. S. Jia, Zhe and D. P. Scarpazza. Dissecting the nvidia volta gpu architecture via microbenchmarking. *arXiv preprint arXiv:1804.06826*, 2018. 6

[45] L. F. Julià and P. Monasse. A critical review of the trifocal tensor estimation. In *Pacific-Rim Symposium on Image and Video Technology*, pages 337–349. Springer, 2017. 5

[46] K. Kanatani, Y. Sugaya, and H. Niitsuma. Triangulation from two views revisited: Hartley-sturm vs. optimal correction. *practice*, 4(5), 2008. 4, 5

[47] J. Kileel. Minimal problems for the calibrated trifocal variety. *SIAM Journal on Applied Algebra and Geometry*, 1(1):575–598, 2017. 5

[48] L. Kneip and S. Lynen. Direct optimization of frame-to-frame rotation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2352–2359, 2013. 2

[49] L. Kneip, R. Siegwart, and M. Pollefeys. Finding the exact rotation between two images independently of the translation. In *European conference on computer vision*, pages 696–709. Springer, 2012. 2

[50] M. Kreuzer and L. Robbiano. *Computational commutative algebra*, volume 1. Springer, 2000. 2

[51] D. J. Kriegman and J. Ponce. Geometric modeling for computer vision. In *Curves and Surfaces in Computer Vision and Graphics II*, volume 1610, pages 250–260. International Society for Optics and Photonics, 1992. 2

[52] J. E. S. F. K. Kuang, Yubin and K. Astrom. Minimal solvers for relative pose with a single unknown radial distortion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 33–40, 2014. 1

[53] Y. Kuang and K. Astrom. Pose estimation with unknown focal length using points, directions and lines. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 529–536, 2013. 8

[54] J. H. M. B. Kukelova, Zuzana and T. Pajdla. Radial distortion homography. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 639–647, 2015. 1

[55] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. In *European Conference on Computer Vision*, pages 302–315. Springer, 2008. 2, 4, 8

[56] Z. Kukelova, J. Kileel, B. Sturmfels, and T. Pajdla. A clever elimination strategy for efficient minimal solvers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4912–4921, 2017. 2

[57] Z. Kukelova and V. Larsson. Radial distortion triangulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9681–9689, 2019. 8

[58] Z. Kukelova, T. Pajdla, and M. Bujnak. Fast and stable algebraic solution to l2 three-view triangulation. In *2013 International Conference on 3D Vision-3DV 2013*, pages 326–333. IEEE, 2013. 5

[59] V. Larsson, K. Astrom, and M. Oskarsson. Efficient solvers for minimal problems by syzygy-based reduction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 820–829, 2017. 1, 2, 3, 5, 8

[60] V. Larsson, M. Oskarsson, K. Astrom, A. Wallis, Z. Kukelova, and T. Pajdla. Beyond Gröbner bases: Basis selection for minimal solvers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3945–3954, 2018. 2

[61] V. Larsson, T. Sattler, Z. Kukelova, and M. Pollefeys. Revisiting radial distortion absolute pose. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1062–1071, 2019. 3

[62] V. Larsson, N. Zobernig, K. Taskin, and M. Pollefeys. Calibration-free structure-from-motion with calibrated radial trifocal tensors. In *European Conference on Computer Vision*, pages 382–399. Springer, 2020. 5, 8

[63] S. Leonardos, R. Tron, and K. Daniilidis. A metric parametrization for trifocal tensors with non-colinear pinholes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 259–267, 2015. 5

[64] B. Li and V. Larsson. GAPS: Generator for automatic polynomial solvers. *arXiv preprint arXiv:2004.11765*, 2020. 2

[65] H. Li and R. Hartley. Five-point motion estimation made easy. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 1, pages 630–633. IEEE, 2006. 4

[66] P. Lindstrom. Triangulation made easy. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1554–1561. IEEE, 2010. 5

[67] A. Mantzaflaris, B. Mourrain, and E. Tsigaridas. On continued fraction expansion of real roots of polynomial systems, complexity and condition numbers. *Theoretical Computer Science*, 412(22):2312–2330, 2011. 2

[68] S. J. Maybank and O. D. Faugeras. A theory of self-calibration of a moving camera. *International journal of computer vision*, 8(2):123–151, 1992. 2

[69] A. Morgan. *Solving polynomial systems using continuation for engineering and scientific problems*. SIAM, 2009. 3, 7

[70] B. Mourrain. Pythagore's dilemma, symbolic-numeric computation, and the border basis method. In *Symbolic-Numeric Computation*, pages 223–243. Springer, 2007. 1, 2

[71] B. Mourrain and P. Trebuchet. Border basis representation of a general quotient algebra. In *Proc. 37th International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 265–272, 2012. 1, 2

[72] G. Nakano. Globally optimal dls method for pnp problem with cayley parameterization. In *BMVC*, pages 78–1, 2015. 8

[73] R. Nath, S. Tomov, and J. Dongarra. An Improved MAGMA GEMM For Fermi Graphics Processing Units. *Int. J. High Perform. Comput. Appl.*, 24(4):511–515, Nov. 2010. 6

[74] D. Nister. An efficient solution to the five-point relative pose problem. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–195. IEEE, 2003. 3, 4

[75] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770, 2004. 1, 3, 8

[76] B. Parisse. A probabilistic and deterministic modular algorithm for computing groebner basis over Q. *arXiv preprint arXiv:1309.4044*, 2013. 7

[77] M. Pharr, editor. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison Wesley, 2005. 6

[78] J. Philip. A non-iterative algorithm for determining all essential matrices corresponding to five point pairs. *The Photogrammetric Record*, 15(88):589–599, 1996. 1, 3

[79] M. Pollefeys. VNL RealNPoly: A solver to compute all the roots of a system of n polynomials in n variables through continuation. *Available at github. com/vxl/vxl/blob/master/core/vnl/algo/source code file vnl rnpoly solve. h*, 1997. 2

[80] J. Pritts, Z. Kukelova, V. Larsson, and O. Chum. Radially-distorted conjugate translations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1993–2001, 2018. 3

[81] L. Quan and Z. Lan. Linear n-point camera pose determination. *IEEE Transactions on pattern analysis and machine intelligence*, 21(8):774–780, 1999. 4

[82] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999. 2

[83] O. Saurer, P. Vasseur, C. Demonceaux, and F. Fraundorfer. A homography formulation to the 3pt plus a common direction relative pose problem. In *Asian Conference on Computer Vision*, pages 288–301. Springer, 2014. 8

[84] A. J. Sommese and C. W. Wampler. *The Numerical solution of systems of polynomials arising in engineering and science*. World Scientific, 2005. 1, 2, 3

[85] H. Stewénius. *Gröbner basis methods for minimal problems in computer vision*. Citeseer, 2005. 2

[86] H. Stewénius, M. Oskarsson, K. Aström, and D. Nistér. Solutions to minimal generalized relative pose problems. In *OMNIVIS 2005*, 2005. 1

[87] L. Svärm, O. Enqvist, F. Kahl, and M. Oskarsson. City-scale localization for cameras with known vertical direction. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1455–1461, 2016. 8

[88] R. N. H. L. Tomov, Stanimire and J. Dongarra. Dense linear algebra solvers for multicore with gpu accelerators. In *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pages 1–8. IEEE, 2010. 6

[89] S. Tomov, J. Dongarra, and M. Baboulin. Towards dense linear algebra for hybrid GPU accelerated manycore systems. *Parallel Computing*, 36(5-6):232–240, June 2010. 6

[90] S. Tomov and J. J. Dongarra. Dense linear algebra for hybrid gpu-based systems. pages 37–55, 2010. 6

[91] J. Verschelde. Algorithm 795: Phcpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Transactions on Mathematical Software (TOMS)*, 25(2):251–276, 1999. 1, 2

[92] C. Wu. P3.5p: Pose estimation with unknown focal length. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2440–2448, 2015. 1

[93] J. Zhao, L. Kneip, Y. He, and J. Ma. Minimal case relative pose computation using ray-point-ray features. *IEEE transactions on pattern analysis and machine intelligence*, 42(5):1176–1190, 2019. 3

[94] Y. Zheng, Y. Kuang, S. Sugimoto, K. Astrom, and M. Okutomi. Revisiting the pnp problem: A fast, general and optimal solution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2344–2351, 2013. 1, 4

[95] Y. Zheng, S. Sugimoto, I. Sato, and M. Okutomi. A general and simple method for camera pose and focal length determination. In *Proceedings of the IEEE/CVF Conference on*

Computer Vision and Pattern Recognition, pages 430–437, 2014. 1