MinBackProp – Backpropagating through Minimal Solvers

Diana Sungatullina[1,2]
diana.sungatullina@cvut.cz

Tomas Pajdla[1]
pajdla@cvut.cz

[1] CIIRC, CTU in Prague, Jugoslavskych partyzanu 1580/3, 16 000, Prague, Czech Republic
[2] FEE, CTU in Prague, Technicka 2, 160 00 Prague, Czech Republic

ABSTRACT

We present an approach to backpropagating through minimal problem solvers in end-to-end neural network training. Traditional methods relying on manually constructed formulas, finite differences, and autograd are laborious, approximate, and unstable for complex minimal problem solvers. We show that using the Implicit function theorem (IFT) to calculate derivatives to backpropagate through the solution of a minimal problem solver is simple, fast, and stable. We compare our approach to (i) using the standard autograd on minimal problem solvers and relate it to existing backpropagation formulas through SVD-based and Eig-based solvers and (ii) implementing the backprop with an existing PyTorch Deep Declarative Networks (DDN) framework [GHC22]. We demonstrate our technique on a toy example of training outlier-rejection weights for 3D point registration and on a real application of training an outlier-rejection and RANSAC sampling network in image matching. Our method provides 100% stability and is 10 times faster compared to autograd, which is unstable and slow, and compared to DDN, which is stable but also slow.

Keywords
minimal solvers, epipolar geometry, backpropagation, outlier removal, implicit function theorem

1 INTRODUCTION

Recently, minimal problem solvers [Nis04, SNKS05, KBP08, LOÅ$^+$18, MVP22] have been incorporated into end-to-end machine learning pipelines in camera localization [BKN$^+$17], image matching [WPS$^+$23], and geometric model estimation by RANSAC [BR19]. The key problem with using minimal problem solvers in end-to-end neural network training is to make them differentiable for backpropagation.

Early attempts to backpropagate through minimal problem solvers used explicit derivative formulas [IVS15, DYH$^+$18, RK18] and finite differences [BKN$^+$17] to compute derivatives for backpropagation. Recent work [WPS$^+$23] proposed computing the derivatives using autograd. These are valid approaches, but they can still be considerably improved. Manual differentiation is laborious and must be done repeatedly for every new problem. Finite differences are approximate and are prone to numerical errors. Using autograd is also limited to relatively simple minimal problem solvers since, for more complex

solvers with large templates [MVP22], differentiating the templates becomes unstable due to, e.g., vanishing of the gradients [BSF94].
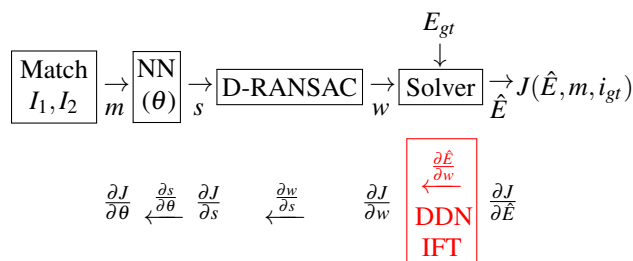

Figure 1: A typical end-to-end training pipeline with a minimal problem Solver [WPS$^+$23] that trains a Neural Network (NN) to predict correct matches. Forward pass: Tentative handcrafted matches m between images I_1, I_2 are clarified using NN [ZGZ$^+$21] parametrized by trained weights θ, and scores s for these matches are computed. Differentiable D-RANSAC selects a minimal data sample w using the scores s; the Solver computes a model $\hat{E}$, which is scored by the loss J using the correct matches $m(i_{gt})$ with the ground truth inlier indicator i_{gt}. The groundtruth E_{gt} is passed to the Solver to choose the closest model. Backward pass: Gradient $\frac{\partial J}{\partial \theta}$ for training weights θ is computed by the chain rule. *The key issue is robustly and efficiently backpropagating through the Solver. We propose to use the Implicit function theorem (IFT) directly or implement the backpropagation via the PyTorch Deep Declarative Network (DDN) machinery [GHC22].*

Let us illustrate using the minimal problem solvers in end-to-end neural network training for calibrated image matching [WPS+23], Fig. 1. First, a for pair of images I_1, I_2 the RootSIFT [AZ12] pre-detected features are used to produce tentative matches m, and the Neural Network $s = \text{NN}(\theta; m)$ computes scores s of the matches m. The network is parameterized by weights θ, the learned pipeline parameters. Next, the differentiable $w = \text{D-RANSAC}(m, s)$ selects a minimal data sample w from m using the s scores. The sample w is passed to the Essential matrix solver $E = \text{Solver}(w)$, which computes up to 10 E candidates and selects the best $\hat{E}$ that is closer to the known ground truth E_{gt}. The Solver can thus be seen as solving a discrete optimization problem consisting of first solving a set of polynomial equations and then choosing the optimal one w.r.t. E_{gt}. Having $\hat{E}$, a projection-based loss $J(\hat{E}, M, i_{gt})$ is calculated from all tentative matches m and the indicator i_{gt} of the known correct (inlier) matches. As a function to be optimized, the pipeline is the composition

$$J(w; E_{gt}, i_{gt}) = J(\hat{E}; m, i_{gt}) \circ \hat{E}(w; E_{gt}) \circ w(s; m) \circ s(\theta; m)$$

where E_{gt}, i_{gt}, and m can be seen as parameters provided by training data.

To learn weights θ, the (transposed) gradient

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \hat{E}} \frac{\partial \hat{E}}{\partial w} \frac{\partial w}{\partial s} \frac{\partial s}{\partial \theta}$$

of J w.r.t. θ has to be calculated. Calculating $\frac{\partial L}{\partial \hat{E}}, \frac{\partial w}{\partial s}, \frac{\partial s}{\partial \theta}$ is straightforward and efficient using autograd. Calculating $\frac{\partial \hat{E}}{\partial w}$ could also be attempted by autograd. We show that it often fails even for a simple minimal problem of computing the calibrated relative camera pose from five image matches [Nis04]. Our main contribution is to show how to calculate $\frac{\partial \hat{E}}{\partial w}$ robustly by the existing PyTorch machinery of Deep Declarative Networks (DDN) [GHC22] and *robustly an efficienlty* using the Implicit function theorem (IFT).

1.1 Motivation

Minimal problem solvers can be complex. Symbolic-numeric minimal problem solvers [EM05, Nis04, LOÅ+18, MVP22] consist of potentially very large templates (formulas combined with the Gauss-Jordan elimination) to construct a matrix whose eigenvectors provide the solutions. The size of the templates is, for systems with a finite number of solutions and more than three unknowns, proportional to a polynomial in D^n, where D is the mean degree of input constraints and n is the number of unknowns [HL11]. In practice, it is not always necessary to construct a full Groebner basis [CLO15], but practical automatic solver generation algorithms [KBP08, LKZ17, MVP22] still

produce very large templates for problems with more variables. See, for example, the sizes of the templates in [MVP22] and the related discussion there.

Therefore, backpropagating through such templates using autograd on the fully expanded computational graph, as in [WPS+23], is generally slower and potentially less stable. Another issue is the backpropagation stability through large eigenvector decompositions [DYH+18]. Backpropagation through constraints at the optimum using the Implicit function theorem (for active KKT constraints in semi-algebraic cases) is much simpler, with linear complexity in the size of the input constraints, and is stable.

1.2 Contribution

We investigate backpropagation methods through minimal problem solvers in end-to-end neural network training. We show that using the Implicit function theorem to calculate derivatives to backpropagate through the solution of a minimal problem solver is simple, fast, and stable. We provide a direct implementation of IFT backpropagation that is stable and fast. We also show how a stable but slower backpropagation can be implemented using the existing PyTorch Deep Declarative Networks framework [GHC22]. This second approach may be helpful for quick testing of functionality before a more efficient IFT backpropagation is implemented.

We compare our approach to using the standard autograd on minimal problem solvers [WPS+23] and relate it to the existing formulas for backpropagating through SVD-based and Eig-based solvers [IVS15, DYH+18].

We demonstrate our technique on toy examples of training outlier-rejection weights for 3D point registration and on a real application of training an outlier-rejection and RANSAC sampling network [WPS+23] in two image matching and camera relative pose computation. Our IFT backpropagation provides 100% stability and is 10 times faster than unstable autograd and DDN.

2 PREVIOUS WORK

The interest in backpropagating through minimal problem solvers started with attempting to develop a differentiable version of RANSAC that could be used in end-to-end learning pipelines.

In seminal work [BKN+17], a differentiable pipeline for camera localization, including RANSAC and PNP solver, has been suggested. The PNP solver was so efficient that it was possible to estimate the derivatives of its output by numerical central differences. We show in Example 1 how to easily estimate the derivatives using the Implicit function theorem.

In [RK18], a method for robust estimation of fundamental matrices embedded in an end-to-end training pipeline was proposed. The goal was to gradually learn

weights for scoring tentative matches to suppress mismatches in a sequence of weighted least-squares problems. Here, the fundamental matrices are computed via SVD. Backpropagating through SVD uses derivatives computed by explicit formulas derived in [IVS15]. Computing derivatives explicitly by formulas is the best approach, but it can be laborious when done manually, as in [IVS15]. We use this example to show in Section 6.1 that our approach based on DDN/IFT provides equally correct results.

Recent work [WPS+23] presents another differentiable pipeline for image matching that includes a differentiable version of RANSAC and minimal problem solvers for fundamental and essential matrix computation. Here, the solvers may use minimal data samples, i.e., 7 and 5 correspondences, respectively, and solve for the matrices using symbolic-numeric algebraic minimal problem solvers, e.g. [Nis04]. To backpropagate through the minimal problem solvers, [WPS+23] uses autograd on the templates and the following Eigendecomposition. As explained above, this is a possible approach for small templates but should be replaced by our approach whenever possible. We show in experiments, Section 6.2, that using autograd often fails even for the relatively simple minimal problem solver of essential matrix estimation. In contrast, our approach based on using the Implicit function theorem delivers 100% stable results and is 10 times faster.

Our approach relies on using the Implicit function theorem [Rud76], which was already used to implement backpropagation for SVD and Eigendecomposition [IVS15, DYH+18] computation blocks of learning pipelines. These approaches manually compute formulas for Jacobians and gradients and implement backward passes for the modules. It is the simplest, fastest, and most robust approach currently possible. It should be used in all final implementations.

To provide a simple but slow error-prone engineering solution for implementing backpropagation through algebraic solvers, we exploit the DDN [GHC22] that implement a framework to backpropagate through algebraic and semi-algebraic optimizers. A similar approach has been suggested in [ZGM+20], but we prefer [GHC22] since it provides PyTorch framework implementation.

2.1 IFT end-to-end learning

The implicit function theorem (IFT) was used in several interesting works related to using geometric optimization in end-to-end learning. None of the works deals with minimal problems, but they are very related to the tasks that are solved with minimal problem solvers, and thus we comment on them here.

In [CPC+20], IFT is used to backpropagate through the least-squares PNP optimization solver for end-to-end learnable geometric vision. Explicit formulas for the Jacobians of the PNP least-squares solvers are derived, and it is shown that they lead to accurate and stable backpropagation. This is an interesting work because it presents a particular generalization of polynomial problems to rational problems but concentrates on one specific geometrical problem.

In [CWW+22], there is a method for backpropagation through a probabilistic PNP. Technically, this approach does not use IFT since the PNP in this formulation is not a solution to equations, but a function providing a distribution over the domain of camera poses. The derivatives of the output are thus readily computable by autograd or by explicit formulas.

In [AK17], the OptNet layer for propagating through optimization problems solved by layers of small quadratic programs is developed using IFT. The aim is to provide an efficient and stable backpropagation that can be implemented on GPU. This is an interesting work since it shows how to backpropagate through an important class of optimization problems. However, it can't address general minimal problems since it only addresses the problems specified as a quadratic program with linear constraints.

3 USING THE IMPLICIT FUNCTION THEOREM

Let us now explain and demonstrate how to compute derivatives of the output of a minimal problem solver using the Implicit function theorem.

Technically, minimal problem solvers generate solutions $\mathbf{x}(\mathbf{a})$ to systems of polynomial equations $f_1(\mathbf{a}),\ldots,f_K(\mathbf{a})$ based on input parameters $\mathbf{a}$. To backpropagate through the minimal problem solvers, it is necessary and sufficient to compute the derivatives $\frac{\partial \mathbf{x}}{\partial \mathbf{a}}$ at each solution $\mathbf{x}$ w.r.t. parameters $\mathbf{a}$.

We now explain how the Implicit function theorem [Rud76] can be used to implement simple, efficient, and numerically stable computation of the derivatives of $\mathbf{x}$ w.r.t. $\mathbf{a}$. Let us start with the formulation of the theorem itself, and then consider a simple example.

Theorem 1 (Differentiating roots of a polynomial system). *Let* $H(\mathbf{a}) = h_1(\mathbf{x},\mathbf{a}),\ldots,h_K(\mathbf{x},\mathbf{a})$ *be a sequence of K complex polynomials in N unknowns* $\mathbf{x} = x_1,\ldots,x_N$ *and M unknowns* $\mathbf{a} = a_1,\ldots,a_M$. *Let* $\mathbf{x}(\mathbf{b})$ *be an isolated multiplicity-one solution to* $H(\mathbf{a})$ *in* $\mathbf{x}$ *for* $\mathbf{a} := \mathbf{b} \in \mathbb{C}^M$. *Then,*

$$\left[\frac{\partial x_n(\mathbf{a})}{\partial a_m}(\mathbf{b})\right] = -\left[\frac{\partial h_k(\mathbf{x},\mathbf{a})}{\partial x_n}(\mathbf{x}(\mathbf{b}),\mathbf{b})\right]^+ \left[\frac{\partial h_k(\mathbf{x},\mathbf{a})}{\partial a_m}(\mathbf{x}(\mathbf{b}),\mathbf{b})\right]$$

with $k = 1,\ldots,K$, $m = 1,\ldots,M$, $n = 1,\ldots,N$, *and* $[A]^+$ *denoting the pseudoinverse [Mey01] of the (Jacobian) matrix* $[A]$.

Proof. Informal statement: The Implicit function theorem is a very standard fact. Here we use it in a special situation when functions are polynomials, since we focus on the Computer Vision problems. The main issue is thus ensuring that the assumptions of the Implicit function theorem are satisfied. For polynomials, which are continuous, infinitely differentiable functions, the assumptions of the Implicit function theorem are satisfied iff the solutions to the polynomials are isolated and have multiplicity one. See Supplementary Material, Section 9 for more details. □

Let us demonstrate the above theorem with an example of the classical computer vision P3P minimal problem of solving the absolute pose of the calibrated camera [FB81].

Example 1. *Consider a general configuration of three 3D points* $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3 \in \mathbb{R}^3$*, their respective calibrated image projections represented by homogeneous coordinates* $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3 \in \mathbb{R}^3$ *and the vector* $\mathbf{x} = [x_1, x_2, x_3]^\top \in \mathbb{C}^3$ *of the depths of the points. Stack all 3D points and image points in a single vector* $\mathbf{a} = [\mathbf{A}_1; \mathbf{A}_2; \mathbf{A}_3; \mathbf{a}_1; \mathbf{a}_2; \mathbf{a}_3] \in \mathbb{R}^{18}$*. The 3D points, their projections, and depths are related by*

$$0 = h_1(\mathbf{x}, \mathbf{a}) = ||\mathbf{A}_1 - \mathbf{A}_2||^2 - ||x_1\mathbf{a}_1 - x_2\mathbf{a}_2||^2$$
$$0 = h_2(\mathbf{x}, \mathbf{a}) = ||\mathbf{A}_2 - \mathbf{A}_3||^2 - ||x_2\mathbf{a}_2 - x_3\mathbf{a}_3||^2$$
$$0 = h_3(\mathbf{x}, \mathbf{a}) = ||\mathbf{A}_3 - \mathbf{A}_1||^2 - ||x_3\mathbf{a}_3 - x_1\mathbf{a}_1||^2$$

The calibrated absolute camera pose computation solves for the depths $\mathbf{x}(\mathbf{a})$ *as functions of the parameters* $\mathbf{a}$*. In this example,* $K = 3$*,* $N = 3$*, and* $M = 18$*. The polynomial system above generically has 8 multiplicity-one solutions* $\mathbf{x}_s(\mathbf{a}), s = 1, \ldots, 8$ *[FB81]. Now, we compute the derivatives of the solutions for a generic parameter vector* $\mathbf{b}$*. Let us first compute the derivatives of the functions* $h_k(\mathbf{x}, \mathbf{a})$ *w.r.t* $\mathbf{x}$ *and* $\mathbf{a}$

$$J_{\mathbf{x}}(\mathbf{x}, \mathbf{a}) = \left[\frac{\partial h_k(\mathbf{x}, \mathbf{a})}{\partial x_n}\right] = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \frac{\partial h_1}{\partial x_3} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \frac{\partial h_2}{\partial x_3} \\ \frac{\partial h_3}{\partial x_1} & \frac{\partial h_3}{\partial x_2} & \frac{\partial h_3}{\partial x_3} \end{bmatrix}(\mathbf{x}, \mathbf{a}) =$$

$$= 2 \begin{bmatrix} -\mathbf{a}_1^\top(x_1\mathbf{a}_1 - x_2\mathbf{a}_2) & \mathbf{a}_2^\top(x_1\mathbf{a}_1 - x_2\mathbf{a}_2) & 0 \\ 0 & -\mathbf{a}_2^\top(x_2\mathbf{a}_2 - x_3\mathbf{a}_3) & \mathbf{a}_3^\top(x_2\mathbf{a}_2 - x_3\mathbf{a}_3) \\ \mathbf{a}_1^\top(x_3\mathbf{a}_3 - x_1\mathbf{a}_1) & 0 & -\mathbf{a}_3^\top(x_3\mathbf{a}_3 - x_1\mathbf{a}_1) \end{bmatrix}$$

$$J_{\mathbf{a}}(\mathbf{x}, \mathbf{a}) = \left[\frac{\partial h_k(\mathbf{x}, \mathbf{a})}{\partial a_m}\right] = \begin{bmatrix} \frac{\partial h_1}{\partial a_1} & \cdots & \frac{\partial h_1}{\partial a_{18}} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_3}{\partial a_1} & \cdots & \frac{\partial h_3}{\partial a_{18}} \end{bmatrix}(\mathbf{x}, \mathbf{a}) = [K_1 | K_2]$$

$$K_1 = 2 \begin{bmatrix} \mathbf{A}_1^\top - \mathbf{A}_2^\top & \mathbf{A}_2^\top - \mathbf{A}_1^\top & 0 \\ 0 & \mathbf{A}_2^\top - \mathbf{A}_3^\top & \mathbf{A}_3^\top - \mathbf{A}_2^\top \\ \mathbf{A}_1^\top - \mathbf{A}_3^\top & 0 & \mathbf{A}_3^\top - \mathbf{A}_1^\top \end{bmatrix}$$

$$K_2 = 2 \begin{bmatrix} -x_1(x_1\mathbf{a}_1 - x_2\mathbf{a}_2) & x_2(x_1\mathbf{a}_1 - x_2\mathbf{a}_2) & 0 \\ 0 & -x_2(x_2\mathbf{a}_2 - x_3\mathbf{a}_3) & x_3(x_2\mathbf{a}_2 - x_3\mathbf{a}_3) \\ x_1(x_3\mathbf{a}_3 - x_1\mathbf{a}_1) & 0 & -x_3(x_3\mathbf{a}_3 - x_1\mathbf{a}_1) \end{bmatrix}$$

Next we evaluate $J_{\mathbf{x}}(\mathbf{x}, \mathbf{a})$ *and* $J_{\mathbf{a}}(\mathbf{x}, \mathbf{a})$ *at a generic parameter vector* $\mathbf{b} = [0, 0, 3, 2, 0, 3, 0, 6, 3, -1/3, -1/3, 1,$ $1/3, -1/3, 1, -1/3, 5/3, 1]$ *and, e.g., the first solution* $\hat{\mathbf{x}}(\mathbf{b}) = [3, 3, 3]$

$$J_{\mathbf{x}}(\hat{\mathbf{x}}(\mathbf{b}), \mathbf{b}) = \begin{bmatrix} -1.33 & -1.33 & 0.00 \\ 0.00 & -5.33 & -21.33 \\ -4.00 & 0.00 & -20.00 \end{bmatrix}$$

$$J_{\mathbf{a}}(\hat{\mathbf{x}}(\mathbf{b}), \mathbf{b}) = \begin{bmatrix} -4 & 0 & 0 & 4 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & -12 & 0 & \ldots & 12 & -36 & 0 \\ 0 & -12 & 0 & 0 & 0 & 0 & \ldots & 0 & -36 & 0 \end{bmatrix}$$

and compute

$$\left[\frac{\partial \hat{x}_n(\mathbf{a})}{\partial a_m}(\mathbf{b})\right] = -J_{\mathbf{x}}(\hat{\mathbf{x}}(\mathbf{b}), \mathbf{b})^{-1} J_{\mathbf{a}}(\hat{\mathbf{x}}(\mathbf{b}), \mathbf{b}) =$$
$$= \begin{bmatrix} 1.66 & 1.33 & 0 & \ldots & 1.25 & 0.24 & 0 \\ 1.33 & -1.33 & 0 & \ldots & -1.25 & -0.24 & 0 \\ -0.33 & 0.33 & 0 & \ldots & -0.25 & -1.75 & 0 \end{bmatrix}.$$

4 IMPLEMENTING MINBACKPROP WITH IFT

The first option to implement the backpropagation for a minimal problem solver is to use the Implicit function theorem in a straightforward way (as in Example 1). This has been done before, e. g., in [IVS15] for SVD and EIG solvers. To use the Implicit function theorem directly, one can construct a system of polynomial equations, compute the derivatives of the system w. r. t. inputs and outputs manually or with the help of a computer algebra system [CSC], and follow the theorem 1 to compute the Jacobian of the output with respect to the input. In case the system of equations is overdetermined, one can write down the Lagrange multipliers method and then use the IFT for the new system.

Applying the Implicit function theorem involves inverting the Jacobian of the polynomial system w. r. t. the output, so before inverting the Jacobian one can check if the matrix is full-rank using SVD. If it is not, one can randomly choose a full-rank Jacobian from the batch to invert.

5 IMPLEMENTING MINBACKPROP WITH DDN

The second option to backpropagate through a minimal problem solver is to use a fully automatic method to compute the derivatives. The Deep Declarative Networks (DDN) framework, introduced in [GHC22], provides a PyTorch implementation allowing backpropagation through optimization solvers, including algebraic, semi-algebraic (i.e., with inequalities), and non-polynomial problems. Next, we explain how to use the DDN framework for implementing backpropagation for minimal problem solvers.

Declarative node: The DDN framework introduces declarative nodes that take input w into the optimal solutions

$$\hat{y}(\mathbf{w}) = \arg\min_{\mathbf{y} \in C} f(\mathbf{w}, \mathbf{y})$$

where C is a constraint set.

Constraint set C**:** For algebraic minimal problem

solvers, the set $C = \{y\colon h_1(y) = 0, \ldots, h_K(y) = 0\}$ is an algebraic variety [CLO15] consisting of a finite number of points. Hence, in this case, the algebraic variety is a finite set of solutions to some system of polynomial equations $H = [h_1, \ldots, h_K]$. Assuming the genericity of data, which is often guaranteed by having noisy measurements, solutions to H have all multiplicity equal to one. Hence it is easy to compute derivatives of the solution to H w.r.t. parameters as demonstrated in Section 3.

Loss f: The low-level loss depends on a particular problem. DDN framework implements backpropagation for a general declarative node and thus expects a loss function $f(w, y)$ with non-degenerate derivatives. For instance, in Section 6.2, thus, one must "invent" such a loss function for minimal problem solvers since they provide exact solutions that always satisfy equations H exactly. A natural choice is to use the sum of squares of the equation residuals from a subset of H, i.e., $f = \sum_{h_i \in G \subseteq H} h_i^2$, such that the derivatives of f are non-degenerate. A possible choice is to use a random linear combination of some h_i's. For instance, in Section 6.2, we use the sum of squares of the linear constraints only to keep f simple and non-degenerate.

It is important to mention that in the pipeline of [WPS$^+$23], the "Solver" not only computes the solutions y to a polynomial system H but also selects the best solution $\hat{y}$ by finding $\hat{y}$ that is closest to the ground-truth y_{gt} in $\|\hat{y} - y_{gt}\|^2$ sense. This discrete optimization step does not influence the derivatives of $\hat{y}$ w.r.t w since, locally, small changes of w do not influence the result of this discrete optimization as small changes to w generically do not change the result of the discrete optimization.

End-to-end training formulation: End-to-end training of a pipeline with a declarative node is in [GHC22] formulated as the following bi-level optimization problem

$$\min_{w} J(w, \hat{y}(w))$$
$$\text{s.t. } \hat{y}(w) \in \arg\min_{y \in C} f(w, y) \qquad (1)$$
$$\text{with } C = \{y\colon h(y) = 0\},$$

where $J(w, y)$ is an upper-level loss, $f(w, y)$ is a low-level loss, $h(y)$ are constraints, w is the input parameters to the declarative node, y is the model to be estimated, and C is a constraint set. The relation $w \mapsto \hat{y}$ is defined as a solution to the low-level optimization problem provided by the declarative node.

Gradients for minimal problem solvers: To minimize the loss $J(w, \hat{y})$ in (1) via the gradient descent,

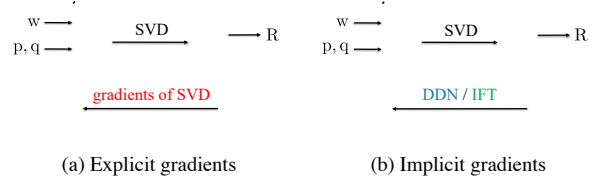(a) Explicit gradients   (b) Implicit gradients

Figure 2: The figure demonstrates the backpropagation for the toy example for the 3D point registration with an outlier. The forward pass is the Kabsch algorithm [Kab76] (SVD) and remains the same for both explicit and implicit methods. The backward pass is performed explicitly via the closed-form gradients of SVD [IVS15] (a) and implicitly, using the Deep Declarative Networks (DDN) [GHC22] and the Implicit function theorem (IFT) (b).

we need to compute $\mathrm{D}J(w, \hat{y})^1$. In our formulation, the loss function $J(w, \hat{y})$ does not depend on w explicitly. Therefore

$$\mathrm{D}J(w, \hat{y}) = \mathrm{D}_y J(w, \hat{y}) \, \mathrm{D}\hat{y}(w) \,. \qquad (2)$$

The gradients $\mathrm{D}_y J(w, \hat{y})$ are the gradients computed before the declarative block and the $\mathrm{D}\hat{y}(w)$ are the gradients through the solution of the low-level optimization problem.

Declarative node specification: The important feature of the DDN framework is that one does not have to provide formulas for the derivatives explicitly. It is enough to specify a declarative node by defining the loss $f(w, y)$ and the constraints $H = [h_1, \ldots, h_K]$. The backward pass based on the derivatives is then automatically computed by the DDN framework. This makes the implementation of backpropagation through minimal problem solvers easy and robust.

6 EXPERIMENTS

In our experiments, we compute the $\mathrm{D}y(w)$ for a solution of a minimal problem in three ways. First, we compute the gradients using the closed-form solution and backpropagate explicitly using autograd. Secondly, we compute the gradients $\mathrm{D}y(w)$ via the Implicit function theorem and backpropagate through a solution by using them directly. Finally, we use the Deep Declarative Networks framework to implement the backpropagation by specifying a suitable optimization problem. Our experiments show that the derivatives of a solution

---

[1] Here we use the notation from [GHC22] where $\mathrm{D}J(w, \hat{y})$ denotes the total derivative $\frac{\mathrm{d}J(w, \hat{y}(w))}{\mathrm{d}w}$ of loss $J(w, \hat{y}(w))$ w.r.t. w when $\hat{y}(w)$ is considered a function of w, $\mathrm{D}_w J(w, \hat{y})$ means partial derivatives $\frac{\partial J(w, \hat{y})}{\partial w}$ of $J(w, \hat{y})$ w.r.t. w when $\hat{y}$ is consider fixed, and $\mathrm{D}_{\hat{y}} J(w, \hat{y})$ means partial derivatives $\frac{\partial J(w, \hat{y})}{\partial \hat{y}}$ of $J(w, \hat{y})$ w.r.t. $\hat{y}$ when w is consider fixed.
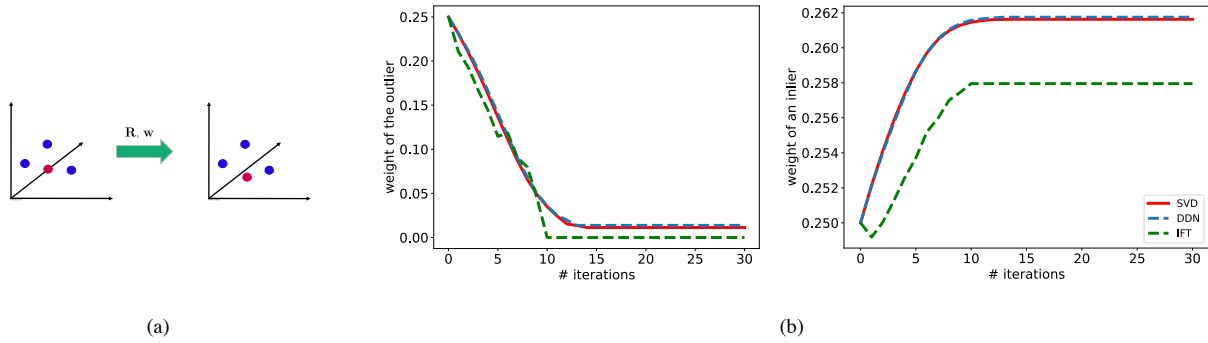
Figure 3: A toy example for the 3D point registration problem with an outlier: (a) visualization of four points before and after rotation R with inlier/outlier weights w, blue points denote inliers, and the magenta point is the outlier; (b) the weight for the outlier (left) and the weight for an inlier (right) during the optimization. The red color indicates explicit computation of the gradients, the blue color denotes backpropagation with the DDN, and the green color is backpropagation with the IFT. One can see that the gradients computed with the DDN and the IFT approximate the gradients of SVD computed in closed form.

to the minimal problem can be computed implicitly; they accurately approximate the derivatives computed explicitly, leading to more stable results.

We consider two types of experiments to show the effectiveness of the proposed method. Firstly, we consider a toy example for the 3D point registration with an outlier. We chose this task since a solution to the minimal problem during the forward pass can be computed using SVD, and the gradients of SVD can be computed both explicitly, using the closed-form solution, and implicitly, using the Implicit function theorem and the Deep Declarative Networks. Secondly, we incorporate our MinBackProp into the existing state-of-the-art epipolar geometry framework, backpropagate through it, and compare the results on the real data.

Our main goal is to show that we obtain equivalent behavior of the whole learning pipeline as the baseline [WPS+23] but in a more stable and efficient way. Note that we do not aim to improve the learning task of [WPS+23] but demonstrate better computational machinery for solving it.

6.1 3D Point Registration with an Outlier

Given two sets P and Q of 3D points, a point $p_i \in P$ corresponds to the point $q_i \in Q$, $i = 1, \dots, N$, where N is the number of points. We want to find a rotation matrix $R \in \mathbb{R}^{3 \times 3}$: $q_i = R p_i$, $\forall i$. Given the ground-truth rotation matrix $R_{\text{true}} \in \mathbb{R}^{3 \times 3}$, we define the weights w_i, $i = 1, \dots, N$, for each correspondence $p_i \leftrightarrow q_i$ to mark inliers and outliers. To find R and $w = [w_1, \dots, w_N]$, let us consider a bi-level optimization problem:

$$\min_{w} J(\hat{R}(w))$$
$$\text{s. t. } \hat{R}(w) \in \arg\min_{R \in SO(3)} f(w, R) \quad (3)$$
$$\text{s. t. } h(R) = 0 \ .$$

Where $J(\hat{R}(w))$ is the upper-lever loss:

$$J(\hat{R}(w)) = \arccos\left(\frac{\text{tr}\left(\hat{R}(w) R_{\text{true}}^T\right) - 1}{2}\right), \quad (4)$$

measuring the angle of the residual rotation [HZ03] and

$$f(P, Q, w, R) = f(w, R) = \frac{1}{N} \sum_{i=1}^{N} w_i \|R p_i - q_i\|_2^2, \quad (5)$$

is the low-lever loss, and $h(R)$ is as the constraint

$$h(R) = R^T R - I = 0 \ . \quad (6)$$

The low-level minimization problem (5) is known as Wahba's problem [Wah65].

Forward pass. During the forward pass, we solve the low-level optimization problem and compute the value of $J(\hat{R}(w))$. The low-level optimization problem is solved by the Kabsch algorithm [Kab76], which includes the computation of SVD of a matrix constructed from coordinates P and Q. The forward pass is the same for both DDN/IFT and SVD layers (Fig. 2).

Backward pass. During the backward pass, we compute all the derivatives with respect to the input and parameters via the chain rule and update the vector w. The backward pass is different for the SVD and the DDN/IFT layers, and we compare these three methods on the toy example below.

A toy example. Let us consider a toy example to compare the gradients computed explicitly and implicitly given a forward pass fixed for both methods. Given four random points in 3D space, we transformed the points with an identity transform R_{true} and corrupted one correspondence, for instance $p_1 \leftrightarrow q_1$. We want to find the rotation matrix R along with inlier/outlier mask w (see Fig. 3 (a)) given the correspondences $p_i \leftrightarrow q_i$.

Method	AUC@5°	AUC@10°	AUC@20°
∇-RANSAC [WPS+23]	0.41	0.45	0.5
MinBackProp DDN	0.4	0.44	0.49
MinBackProp IFT	0.41	0.45	0.5

Table 1: The average AUC scores for essential matrix estimation of ∇-RANSAC [WPS+23] and MinBackProp (ours) over 12 scenes of the PhotoTourism dataset [JMM+20], under different thresholds. Our method shows the same result as ∇-RANSAC.

We start with the uniform distribution of weights w and initialize them with $1/N$. The developments of the values of w for an inlier and the outlier during the optimization are shown in Fig. 3 (b). We optimize the weights w using the gradient descent with the learning rate of 0.1 for 30 iterations. The values of w for the outlier decrease to 0 and the values of w for inliers increase. One can see that the gradients calculated implicitly approximate the gradients of SVD computed in closed form well.

One can find one more toy example for the fundamental matrix estimation (8-point) in the Supplementary Material, section 10.

6.2 Training Outlier Detection in Epipolar Geometry Estimation

Our second experiment is incorporating our MinBackProp into the existing state-of-the-art epipolar geometry pipeline with real data. For that purpose, we consider a fully-differentiable framework ∇-RANSAC [WPS+23] that exploits the 5-point algorithm [Nis04] for essential matrix estimation and evaluates it on the RootSIFT [AZ12] features of the PhotoTourism dataset [JMM+20]. Figure 1 illustrates the architecture of ∇-RANSAC and the way how we integrate our MinBackProp in it. We adopt all the formulations and processing of [WPS+23], except for the backpropagation through the minimal problem solver (red rectangle in Fig. 1).

Given a pair of images I_1 and I_2 and a set of tentative correspondences $m = [q_i, \tilde{q}_i]$ computed from the RootSIFT features, the inlier indicator i_{gt}, and the ground truth essential matrix E_{gt}. The goal is to train a Neural Network (NN) [ZGZ+21] with parameters θ that predicts importance scores s during inference. In Fig. 1, D-RANSAC indicates differentiable Gumbel Softmax sampler, w denotes a minimal sample used as input into the Solver, and J is the loss function.

In accordance with eq. (1), we formulate our bilevel optimization problem as follows. The upper-level objective J is the symmetric epipolar distance over the inlier set i_{gt} of matches $m_i = [q_i, \tilde{q}_i]$, $\hat{E}$ is the predicted model

Method	stable runs (%)	backward time (s)
∇-RANSAC [WPS+23]	20	34.4
∇-RANSAC*	30	34.5
MinBackProp DDN (ours)	**100**	37.6
MinBackProp IFT (ours)	**100**	**3.6**

Table 2: Stability metrics for 1K training of the baseline ∇-RANSAC [WPS+23] and our MinBackProp methods for the essential matrix estimation. ∇-RANSAC* denotes implementation of ∇-RANSAC [WPS+23] without dropping non-real values in the solver during the forward pass. Our MinBackProp is much more stable compared to the ∇-RANSAC and ∇-RANSAC* and MinBackProp IFT ten times faster than the baseline.

to be estimated, and $\tilde{l}_i = \hat{E} q_i$, $l_i = \hat{E}^T \tilde{q}_i$ are the epipolar lines

$$J(\hat{E}, m, i_{gt}) = \frac{1}{|i_{gt}|} \sum_{i \in i_{gt}} \left(\frac{1}{l_{i1}^2 + l_{i2}^2} + \frac{1}{\tilde{l}_{i1}^2 + \tilde{l}_{i2}^2} \right) \left(\tilde{q}_i^T \hat{E} q_i \right)^2,$$
(7)

while the low-level objective is the "invented" loss consisting of the algebraic error of the constraints (which is always evaluated to zero but provides non-singular Jacobians, see the discussion in section 5)

$$\hat{E}(w) \in \arg\min_{E \in \mathbb{R}^{3\times3}} \frac{1}{5} \sum_{i=1}^{5} \left(\tilde{q}_i^T E q_i \right)^2$$
$$\text{s.t. } 2 E E^T E - \text{tr}\left(E E^T\right) E = 0,$$
$$\|E\|^2 = 1,$$
(8)

where $w_i = [q_i, \tilde{q}_i]$ is a pair of matches from a minimal sample normalized by the intrinsic matrix pixel coordinates, and E is an essential matrix. The same equations eq. (8) are used to construct the polynomial system for the Implicit function theorem and compute the derivatives w.r.t. $\hat{E}$ and $[q_i, \tilde{q}_i]$ and the Jacobian $\frac{\partial \hat{E}}{\partial w}$, respectively. See Supplementary Material, section 11 for more details.

As in the section 6.1, we retain the forward pass for the MinBackProp unchanged. However, we modify the computation of gradients during the backward pass. Instead of computing the gradients explicitly by the autograd as in ∇-RANSAC, we exploit the Deep Declarative Networks and the Implicit function theorem.

For training, as in ∇-RANSAC, we use correspondences of the St. Peter's Square scene of the PhotoTourism dataset [JMM+20], consisting of 4950 image pairs, with split 3 : 1 into training and validation, while the rest 12 scenes remain for testing.

To show the effect of our method, we perform two types of experiments. First, we want to show that our method does not decrease the baseline quality. Secondly, we want to show that computing the gradients via the Implicit function theorem is more stable and does not cause runtime errors during the backward pass when autograd tries to invert a singular matrix.

For the first experiment, the testing protocol is as follows: the scores s are predicted by the trained NN, and the model E is estimated by MAGSAC++ [BNIM20]. Table 1 shows the Area Under the Recall curve (AUC), AUC scores are averaged over 12 testing scenes for different thresholds. The essential matrix E is decomposed into R and t using SVD and maximum error between R and R_{gt} and t and t_{gt} is reported. We train all three approaches for 10 epochs. One can see in Table 1 that our MinBackProp obtained the same scores as the ∇-RANSAC, showing that it does not decrease the quality of the baseline.

The second experiment aims to demonstrate the stability of the proposed MinBackProp compared to the ∇-RANSAC. For this purpose, we perform the following experiment. Both methods are trained for 10 epochs, and 10 random runs are performed. We measure the percentage of runs which finished 10 epochs training without any runtime errors during the backpropagation. The results are reported in Table 2. Our method is 100% stable compared to the baseline, which is 30% stable, and it is also $\times 10$ faster. ∇-RANSAC* denotes the implementation of the original ∇-RANSAC without dropping non-real solutions during the forward pass of the solver since to backpropagate the gradients with the DDN, we need to find at least one solution to the optimization problem, so in case the solver can not find any real solution, we return the one from the previous iteration. We also measured the stability of the ∇-RANSAC* to be sure that this minor change does not contribute to the stability of our method.

All the experiments were conducted on NVIDIA GeForce GTX 1080 Ti with CUDA 12.2 and Pytorch 1.12.1 with Adam optimizer [KB15], learning rate 10^{-4} and batch size 32.

7 CONCLUSION

We have presented a new practical approach to backpropagating through minimal problem solvers. Our MinBackProp allows backpropagation either using the Deep Declarative Networks machinery, which brings stability and easy use, or using the Implicit function theorem directly, which on top of stability, also speeds up significantly the backward pass. Through synthetic examples, we have shown the applicability of our method on a wide variety of tasks. Furthermore, we compared MinBackProp against the state-of-the-art relative pose estimation approach from [WPS$^+$23] and have shown 100% stability compared to $70-80\%$ failure rate of the autograd approach, while speeding up the computation 10 times. Our method opens up a promising direction for efficient backpropagation through hard minimal problems.

See Supplementary Material for additional technical details. The code is available at https://github.com/disungatullina/MinBackProp .

8 REFERENCES

[AK17] Brandon Amos and J. Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 136–145. PMLR, 2017. 3

[AZ12] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2911–2918, 2012. 2, 7

[BKN$^+$17] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac – differentiable ransac for camera localization. pages 2492–2500, 07 2017. 1, 2

[BNIM20] Daniel Barath, Jana Noskova, Maksym Ivashechkin, and Jiri Matas. MAGSAC++, a fast, reliable and accurate robust estimator. In *Conference on Computer Vision and Pattern Recognition*, 2020. 8

[BR19] Eric Brachmann and Carsten Rother. Neural-guided ransac: Learning where to sample model hypotheses. pages 4321–4330, 10 2019. 1

[BSF94] Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2):157–166, 1994. 1

[CLO98] David Cox, John Little, and Donald O'Shea. *Using Algebraic Geometry*. Springer, 1998. 1

[CLO15] David A. Cox, John Little, and Donald O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 2015. 2, 5

[CPC+20] Bo Chen, Álvaro Parra, Jiewei Cao, Nan Li, and Tat-Jun Chin. End-to-end learnable geometric vision by backpropagating pnp optimization. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 8097–8106. Computer Vision Foundation / IEEE, 2020. 3

[CSC] Ltd. Cybernet Systems Co. Maple. http://www.maplesoft.com/products/maple/. 4

[CWW+22] Hansheng Chen, Pichao Wang, Fan Wang, Wei Tian, Lu Xiong, and Hao Li. Epropnp: Generalized end-to-end probabilistic perspective-n-points for monocular object pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 2771–2780. IEEE, 2022. 3

[DYH+18] Zheng Dang, Kwang Moo Yi, Yinlin Hu, Fei Wang, Pascal Fua, and Mathieu Salzmann. Eigendecomposition-free training of deep networks with zero eigenvalue-based losses. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part V*, volume 11209 of *Lecture Notes in Computer Science*, pages 792–807. Springer, 2018. 1, 2, 3

[EM05] Mohamed Elkadi and Bernard Mourrain. *Symbolic-numeric methods for solving polynomial equations and applications*, pages 125–168. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. 2

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981. 4

[GHC22] Stephen Gould, Richard Hartley, and Dylan Campbell. Deep declarative networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):3988–4004, 2022. 1, 2, 3, 4, 5

[Har95] R. I. Hartley. In defence of the 8-point algorithm. In *Proceedings of the Fifth International Conference on Computer Vision*, ICCV '95, page 1064, USA, 1995. IEEE Computer Society. 2

[HL11] AMIR HASHEMI and DANIEL LAZARD. Sharper complexity bounds for zero-dimensional groebner bases and polynomial system solving. *International Journal of Algebra and Computation*, 21(05):703–713, 2011. 2

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge, 2nd edition, 2003. 6

[IVS15] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Matrix back-propagation for deep networks with structured layers. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2965–2973, 2015. 1, 2, 3, 4, 5

[JMM+20] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image matching across wide baselines: From paper to practice. *International Journal of Computer Vision*, 2020. 7, 3

[Kab76] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923, Sep 1976. 5, 6

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 8

[KBP08] Zuzana Kukelova, Martin Bujnak, and Tomas Pajdla. Automatic generator of minimal problem solvers. In *European Conference on Computer Vision (ECCV)*, 2008. 1, 2

[LKZ17] 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. 2

[LOÅ+18] 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. 1, 2

[Mey01] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. 3, 1

[MVP22] Evgeniy Martyushev, Jana Vráblíková,

and Tomás Pajdla. Optimizing elimination templates by greedy parameter search. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 15733–15743. IEEE, 2022. 1, 2

[Nis04]   David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:756–770, 2004. 1, 2, 3, 7

[RK18]   Rene Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *The European Conference on Computer Vision (ECCV)*, 2018. 1, 2

[Rud76]   Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 1976. 3, 1

[SNKS05]   Henrik Stewénius, David Nistér, Fredrik Kahl, and Frederik Schaffalitzky. A minimal solution for relative pose with unknown focal length. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*, pages 789–794. IEEE Computer Society, 2005. 1

[SR13]   I.R. Shafarevich and M. Reid. *Basic Algebraic Geometry 1: Varieties in Projective Space*. SpringerLink : Bücher. Springer Berlin Heidelberg, 2013. 1

[Wah65]   Grace Wahba. A least squares estimate of satellite attitude. *SIAM Review*, 7(3):409–409, 1965. 6

[WPS+23]   Tong Wei, Yash Patel, Alexander Shekhovtsov, Jiri Matas, and Daniel Barath. Generalized differentiable ransac. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17649–17660, October 2023. 1, 2, 3, 5, 6, 7, 8

[ZGM+20]   Qianggong Zhang, Yanyang Gu, Mateusz Michalkiewicz, Mahsa Baktashmotlagh, and Anders P. Eriksson. Implicitly defined layers in neural networks. *CoRR*, abs/2003.01822, 2020. 3

[ZGZ+21]   Chen Zhao, Yixiao Ge, Feng Zhu, Rui Zhao, Hongsheng Li, and Mathieu Salzmann. Progressive correspondence pruning by consensus learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.*, 2021. 1, 7