Progressive Batching for Efficient Non-linear Least Squares

Huu Le[1 [0000−0001−7562−7180]], Christopher Zach[1 [0000−0003−2840−6187]], Edward Rosten[2 [0000−0001−8675−4230]], and Oliver J. Woodford[2 [0000−0002−4202−4946]]

[1] Chalmers University, Sweden **
[2] Snap, Inc., London & Santa Monica

Abstract. Non-linear least squares solvers are used across a broad range of offline and real-time model fitting problems. Most improvements of the basic Gauss-Newton algorithm tackle convergence guarantees or leverage the sparsity of the underlying problem structure for computational speedup. With the success of deep learning methods leveraging large datasets, stochastic optimization methods received recently a lot of attention. Our work borrows ideas from both stochastic machine learning and statistics, and we present an approach for non-linear least-squares that guarantees convergence while at the same time significantly reduces the required amount of computation. Empirical results show that our proposed method achieves competitive convergence rates compared to traditional second-order approaches on common computer vision problems, such as image alignment and essential matrix estimation, with very large numbers of residuals.

1 Introduction

Non-linear least squares (NLLS) solvers [1] are the optimizers of choice for many computer vision model estimation and fitting tasks [2], including photometric image alignment [3], essential matrix estimation [2] and bundle adjustment [4]. Fast convergence due to the second-order gradient model, and a simple, efficient implementation due to Gauss' approximation of the Hessian, make it a highly effective tool for these tasks. Nevertheless, the (non-asymptotic) computational efficiency of these methods can significantly impact the overall performance of a vision system, and the need to run such tasks in real-time, at video frame-rate, for applications such as Augmented Reality, leads to ongoing research to improve NNLS solvers.

Standard NLLS solvers such as the Gauss-Newton (GN) [5] or Levenberg-Marquardt (LM) method [6,7] evaluate all residuals and their Jacobians (first derivatives of the residual function) at every iteration. Analogous to large-scale machine learning, utilizing all the data available to a problem can therefore substantially and unnecessarily slow down the optimization process. No improvements in solver efficiency have seen widespread adoption for model fitting, to

** This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

address this problem. In practice, systems are engineered to pre-select a sparse set of data to avoid it, requiring some tuning on the part of the implementer to ensure that enough data is included for robustness and accuracy, while not too much is included, to keep computation time within budget. These design decisions, made at "compile time", do not then adapt to the unknown and variable circumstances encountered at run time.

Inspired by the stochastic methods used to accelerate large-scale optimization problems in machine learning, we introduce a stochastic NLLS optimizer that significantly reduces both computation time and the previously linear relationship between number of residuals and computation time, at no cost to accuracy. Our method has the following novel features:

- A stochastic, variable batch size approach for non-linear least squares that can be easily integrated into existing NLLS solvers for model fitting applications.
- A statistical test to determine the acceptance of an update step computed from a batch, without evaluating all residuals, that also progressively increases the batch size.
- Guaranteed convergence to a local minimum of the original problem, since all residuals are automatically included in the final iterations.

By adjusting the batch size at run time, according to a reasoned, statistical test, our algorithm is able to invest more computational resources when required, and less when not. This avoids the need to tightly tune the number of residuals at compile time, and can therefore lead to more accurate performance as a result.

We evaluate our method on a number of two-view geometry problems involving geometric and photometric residuals, such as essential matrix estimation and homography fitting, with promising results[3]. In particular, we empirically show that our new approach has much faster convergence rates compared to conventional approaches.

2 Related Work

2.1 Non-linear least squares

Fully second-order optimizers, such as NLLS methods, benefit from both the automatic choice of the step size and from modelling the dependency between variables, which results in faster convergence (rates) than first order and even quasi-Newton methods. Gauss replaced the Hessian of Newton's method with an approximation for least squares costs in 1809, creating the original NLLS solver, the Gauss-Newton method [5]. Since the Gauss-Newton method does not guarantee convergence, the Levenberg-Marquardt algorithm [6,7] extends Gauss-Newton with back-tracking (i.e. conditional acceptance of new iterates) and diagonal damping of the (approximate) Hessian. More recently, a further modification of the Gauss-Newton method, variable projection (VarPro [8,9,10]),

[3] Our source code is available at `https://github.com/intellhave/ProBLM`

has been proposed for a particular class of separable problems, resulting in wider convergence basins to reach a global solution.

Since many model fitting tasks are solved using NLLS, several acceleration techniques have been developed to address problems of scale and real-time operation. However, these techniques are not generic, but instead exploit task specific properties. For example, certain image alignment tasks have been accelerated by the inverse compositional algorithm [3], which computes Jacobians once only, in the reference frame, or by learning a hyperplane approximation to the pseudo-inverse [11]. Truncated and inexact Newton methods [5] typically use iterative solvers such as conjugate gradients (CG) to approximately solve the linear system. These can converge in less time on larger scale problems, such as bundle adjustment (BA), especially when efficient, BA specific preconditioners are used [12]. In a framework such as RANSAC [13], convergence for each subproblem is not required so significant speedups are available by allowing a reduced probability of convergence [14].

Huge gains in efficiency are available if the problem exhibits a conditional independence structure. Bundle adjustment and related bipartite problem instances use techniques such as the Schur complement [4] (or more generally column reordering [15]) to reduce the size of the linear system to be solved in each iteration. Other linear systems have linear time solvers: Kalman smoothing [16] is a special case of using belief propagation [17] to solve linear least squares and such techniques can be applied to non-linear, robust least squares over tree structures [18].

2.2 Stochastic methods

Stochastic first order methods [19,20], which are now common for large-scale optimization in machine learning, compute approximate (first order) gradients using subsets of the total cost, called batches, thereby significantly accelerating optimization. The randomness of the method (and the intermediate solutions) requires certain provisions in order to obtain guarantees on convergence. Due to their stochastic nature these methods empirically return solutions located in "wide" valleys [21,22].

In addition to first-order methods stochastic second-order one have also been investigated (e.g. [23,24,25,26]). One main motivation to research stochastic second-order methods is to overcome some shortcomings of stochastic first-order methods such as step size selection by introducing curvature (2nd-order) information of the objective. Due to the scale of problems tackled, many of these proposed algorithms are based on the L-BFGS method [27,28], which combines line search with low-rank approximations for the Hessian. The main technical difference between stochastic first-order and second-order methods is that the update direction in stochastic gradient methods is an unbiased estimate of the true descent direction, which is generally not the case for stochastic second-order updates. Convergence of stochastic methods relies on controlling the variance of the iterates, and consequently requires e.g. diminishing step sizes in stochastic first order methods [19,29] or e.g. shrinking trust region radii [26] for a second

order method. Without diminishing variances of the iterates, it is only possible to return approximate solutions [30]. Hence, using a proper stochastic method to minimize an NNLS instance over a finite number of residuals will never fully benefit from the fast convergence rates of 2nd order methods.

A number of recent proposals for stochastic methods use batches with an adaptive (or dynamically adjusted) batch size. This is in particular of interest in the second-order setting, since increasing the batch is one option to control the variance of the iterates (another option being reducing the step size). [31] and [32] propose schemes to adjust the batch size dynamically, which mostly address the first-order setup. Bollapragada et al. [24] propose a randomized qausi-Newton method, that is designed to have similar benefits as the stochastic gradient method. The method uses progressively larger sets of residuals, and the determination of the sample size is strongly linked with the underlying L-BFGS algorithm. A trust region method with adaptive batch size is described in [33], where the decision to accept a new iterate is based on the full objective (over all residuals), rather than the batch subset, limiting the benefits of this method. Similarly, [34] and [35] propose stochastic second-order methods that build on stochastic approximations of the Hessian but require computation of the full gradient (which was later relaxed in [36]).

The stochastic approaches above target optimizations with a large number of model parameters, which each depend on a large number of residuals. The scale of such problems does usually not permit the direct application of standard NLLS solvers. In many model fitting problems in computer vision, the number of variables appearing in the (Schur-complement reduced) linear system is small, making NLLS a feasible option. L-BFGS [37] (and any quasi-Newton method) has a disadvantage compared to Gauss-Newton derivatives for NNLS problem instances, since the NNLS objective is near quadratic when close to a local minimum. Thus, L-BFGS is not favoured in these applications, due to its slower convergence rate than NLLS (which is also empirically verified in Section 5). We note that the progressive batching of residuals introduced in this work can in principle also be applied to any local optimization method that utilizes backtracking to conditionally accept new iterates.

3 Background

3.1 Problem Formulation

Our work tackles the following NLLS problem:

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} f_i(\boldsymbol{\theta}), \quad \text{where} \quad f_i(\boldsymbol{\theta}) = \|\mathbf{r}_i(\boldsymbol{\theta})\|_2^2 \tag{1}$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ is the vector containing the desired parameters, N specifies the number of available measurements (residuals), and $\mathbf{r}_i : \mathbb{R}^d \mapsto \mathbb{R}^p$ is the function that computes the residual vector (of length p) of the i-th measurement.

It is worth noting that, while we start by introducing the standard NLLS formulation as per (1), our proposed method is directly applicable to robust parameter estimation problems through the use of methods such as Iteratively Reweighted Least Squares (IRLS) [38], since each step of IRLS can be reformulated as a special instance of (1). More specifically, with the use of a robust kernel ψ to penalize outlying residuals, the robust parameter estimation is defined as,

$$\min_{\boldsymbol{\theta}} \sum\nolimits_{i=1}^{N} g_i(\boldsymbol{\theta}), \quad \text{where} \quad g_i(\boldsymbol{\theta}) = \psi(\|\mathbf{r}_i(\boldsymbol{\theta})\|). \tag{2}$$

3.2 Levenberg-Marquardt Method

While our progressive batching approach is applicable to any second-order algorithm, here we employ Levenberg-Marquardt (LM) as the reference solver, as it is the most widely used method in a number of computer vision problems. In this section we briefly review LM and introduce some notations which are also later used throughout the paper.

At the t-th iteration, denote the current solution as $\boldsymbol{\theta}^{(t)}$ and $\{\mathbf{r}_i(\boldsymbol{\theta}^{(t)})\}_{i=1}^{N}$ as the set of residual vectors for all measurements. The LM algorithm involves computing the set of Jacobians matrices $\{\mathbf{J}_i^{(t)} \in \mathbb{R}^{p\times d}\}$, where, using $\mathbf{r}_i^{(t)}$ as shorthand for $\mathbf{r}_i(\boldsymbol{\theta}^{(t)})$,

$$\mathbf{J}_i^{(t)} = \begin{bmatrix} \frac{\partial \mathbf{r}_i^{(t)}}{\partial \boldsymbol{\theta}_1^{(t)}} & \frac{\partial \mathbf{r}_i^{(t)}}{\partial \boldsymbol{\theta}_2^{(t)}} & \cdots & \frac{\partial \mathbf{r}_i^{(t)}}{\partial \boldsymbol{\theta}_d^{(t)}} \end{bmatrix}. \tag{3}$$

Based on the computed Jacobian matrices and the residual vectors, the gradient vector $\mathbf{g}^{(t)}$ and approximate Hessian matrix $\mathbf{H}^{(t)}$ are defined as follows:

$$\mathbf{g}^{(t)} \coloneqq \sum\nolimits_{i=1}^{N} (\mathbf{J}_i^{(t)})^T \mathbf{r}_i^{(t)}, \qquad \mathbf{H}^{(t)} \coloneqq \sum\nolimits_{i=1}^{N} (\mathbf{J}_i^{(t)})^T \mathbf{J}_i^{(t)}, \tag{4}$$

where $(.)^T$ denotes the matrix transpose operation. Given $\mathbf{g}^{(t)}$ and $\mathbf{H}^{(t)}$, LM computes the step $\Delta\boldsymbol{\theta}$ by solving

$$\Delta\boldsymbol{\theta}^{(t)} \leftarrow -(\mathbf{H}^{(t)} + \lambda \mathbf{I})^{-1} \mathbf{g}^{(t)}, \tag{5}$$

where λ is the damping parameter that is modified after each iteration depending on the outcome of the iteration. In particular, if the computed step leads to a reduction in the objective function, i.e., $f(\boldsymbol{\theta}^{(t)} + \Delta\boldsymbol{\theta}^{(t)}) < f(\boldsymbol{\theta}^{(t)})$, the step is accepted and $\boldsymbol{\theta}$ is updated by setting $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^{t} + \Delta\boldsymbol{\theta}$, while the damping λ is decreased by some factor. On the other hand, if $f(\boldsymbol{\theta}^{(t)} + \Delta\boldsymbol{\theta}^{(t)}) \geq f(\boldsymbol{\theta}^{(t)})$, $\Delta\boldsymbol{\theta}^{(t)}$ is rejected, λ is increased accordingly and (5) is recomputed; this is repeated until the cost is reduced.

4 Proposed Algorithm

In this section, we describe our proposed algorithm that is computationally cheaper than conventional second-order approaches. The main idea behind our

algorithm is that, instead of computing the Jacobians of all residuals (as described in Sec. 3.2), we utilize only a small fraction of measurements to approximate the gradient and Hessian, from which the update step is computed.

Let $\mathcal{S}^{(t)} \subseteq \{1 \dots N\}$ denote the subset of residual indices that are used at the t-th iteration (note that the set of subsampled indices can remain the same throughout many iterations, i.e., $\mathcal{S}^{t_0} = \cdots = \mathcal{S}^{t}(t_0 \leq t)$). For the ease of notation, we use $\mathcal{S}$ to specify the subset being used at the current iteration. Given a subset $\mathcal{S}$, analogous to (4), we use the notation $\mathbf{g}_\mathcal{S}$ and $\mathbf{H}_\mathcal{S}$ to denote the approximate gradient and Hessian obtained by only using residuals in the subset $\mathcal{S}^{(t)}$, i.e.

$$\mathbf{g}_\mathcal{S}^{(t)} \coloneqq \sum\nolimits_{i \in \mathcal{S}^{(t)}} \mathbf{J}_i^{(t)} \mathbf{r}_i^{(t)} \qquad\qquad \mathbf{H}_\mathcal{S}^{(t)} \coloneqq \sum\nolimits_{i \in \mathcal{S}^{(t)}} (\mathbf{J}_i^{(t)})^T \mathbf{J}_i^{(t)}. \tag{6}$$

We also define the subset cost $f_\mathcal{S}^{(t)}$ as $f_\mathcal{S}^{(t)} \coloneqq \sum_{i \in \mathcal{S}} f_i(\boldsymbol{\theta})$. Similar to (5), the approximate update step is denoted by $\Delta\hat{\boldsymbol{\theta}}_{\mathcal{S}^{(t)}}$, and is computed by

$$\Delta\boldsymbol{\theta}_\mathcal{S}^{(t)} \leftarrow -(\mathbf{H}_\mathcal{S}^{(t)} + \lambda \mathbf{I})^{-1} \mathbf{g}_\mathcal{S}^{(t)}. \tag{7}$$

Note that depending on the characteristic of $\mathcal{S}^{(t)}$, $\mathbf{g}_\mathcal{S}^{(t)}$ and $\mathbf{H}_\mathcal{S}^{(t)}$ can be very far from the true $\mathbf{g}^{(t)}$ and $\mathbf{H}^{(t)}$, respectively. As a consequence, the update step $\Delta\boldsymbol{\theta}_\mathcal{S}^{(t)}$ computed from (7), despite resulting in a reduction in $f_\mathcal{S}^{(t)}$, may lead to an increase in the original cost $f^{(t)}$. However, as the number of measurements can be very large, computing the whole set of residuals at each iteration to determine the acceptance of $\Delta\boldsymbol{\theta}_{\mathcal{S}^{(t)}}$ is still very costly. Hence, we employ statistical approaches. Specifically, we only accept $\Delta\boldsymbol{\theta}_\mathcal{S}^{(t)}$ if, with a probability not less than $1 - \delta$ $(0 < \delta < 1)$, a reduction in $f_\mathcal{S}^{(t)}$ also leads to a reduction in the true cost $f^{(t)}$. More details are discussed in the following sections.

4.1 A Probabilistic Test of Sufficient Reduction

We now introduce a method to quickly determine if an update step $\Delta\boldsymbol{\theta}_\mathcal{S}^{(t)}$ obtained from (7) also leads to a sufficient reduction in the original cost with a high probability. To begin, let us define $\boldsymbol{\theta}^{(t+1)} \coloneqq \boldsymbol{\theta}^{(t)} + \Delta\boldsymbol{\theta}_\mathcal{S}^{(t)}$, and denote by

$$X_i = f_i(\boldsymbol{\theta}^{(t+1)}) - f_i(\boldsymbol{\theta}^{(t)}) \tag{8}$$

the change of the i-th residual. We convert X_i to a random variable by drawing the index i from a uniform distribution over $\{1, \dots, N\}$. Taking K random indices (which form the subset $\mathcal{S}$) yields the random variables $(Y_1, \dots, Y_K)$, where

$$Y_k = X_{i_k} \qquad i_k \sim U\{1, \dots, N\}. \tag{9}$$

We can observe that the expectation

$$\mathbb{E}[Y_k] = f(\boldsymbol{\theta}^{(t+1)}) - f(\boldsymbol{\theta}^{(t)}) = \sum\nolimits_i \left(f_i(\boldsymbol{\theta}^{(t+1)}) - f_i(\boldsymbol{\theta}^{(t)}) \right), \tag{10}$$

represents the total change of the true objective, and at each iteration we are interested in finding $\boldsymbol{\theta}^{(t+1)}$ such that $\mathbb{E}[Y_k] < 0$. To obtain a lower bound for the random variables (which will be useful for the test introduced later), we can clamp Y_k to a one-sided range $[a, \infty)$ by introducing $Z_k := \max(a, Y_k)$. It can be noted that $\mathbb{E}[Y_k] \leq \mathbb{E}[Z_k]$ and therefore $P(\mathbb{E}[Y_k] \geq 0) \leq P(\mathbb{E}[Z_k] \geq 0)$, hence we can safely use $\mathbb{E}[Z_k]$ as a proxy to evaluate $\mathbb{E}[Y_k]$.

We introduce $S_K := \sum_{k=1}^{K} Z_k$, representing (an upper bound to) the observed reduction, i.e. $S_K \geq f_{\mathcal{S}}(\boldsymbol{\theta}^{(t+1)}) - f_{\mathcal{S}}(\boldsymbol{\theta}^{(t)})$. Recall that, during optimization, S_K is the only information available to the algorithm, while our real information of interest is the expectation $\mathbb{E}[Z_k]$. Therefore, it is necessary to establish the relation between $\mathbb{E}[Z_k]$ and S_K. Assume that $S_K < 0$ (i.e., the update step leads to a reduction in the observed cost), given a probability $0 < \delta < 1$, and a scalar $0 \leq \alpha < 1$, we are interested in the following criterion,

$$P\left(\mathbb{E}[S_K] \leq \alpha S_K\right) \geq 1 - \delta, \tag{11}$$

indicating whether the true cost is also reduced by at least a fraction α of the observed reduction S_K with probability $1-\delta$. Using Hoeffding's inequality [39][4], we obtain

$$P(\mathbb{E}[S_K] \geq \alpha S_K) = P\left(S_K - \mathbb{E}[S_K] \leq (1-\alpha) S_K\right) \leq \exp\left(-\tfrac{2(1-\alpha)^2 S_K^2}{K(b-a)^2}\right), \tag{12}$$

where $b \in \mathbb{R}$ is the upper bound of the random variables Z_k ($Z_k \leq b \; \forall k$). While the lower bound a can be freely chosen, computing b is often more involved in practice (we will discuss several options to choose b in the following section).[5]

In order for (11) to hold, we require the r.h.s. of (12) to be upper-bounded by a user-specified confidence level $\delta \in (0, 1)$, i.e., $\exp\left(-\tfrac{2(1-\alpha)^2 S_K^2}{K(b-a)^2}\right) \leq \delta$, which leads to the condition

$$S_K \leq -\frac{b-a}{1-\alpha} \cdot \sqrt{\frac{-K \log \delta}{2}} \qquad \big(\leq 0\big). \tag{13}$$

Thus, if the condition (13) is satisfied, we can confidently accept the step computed from the subset $\mathcal{S}$. More specifically, based on S_K, the following steps are applied for the LM iterations on the subset:

1. $S_K \geq 0$: increase λ (e.g. $\lambda \leftarrow 10\lambda$), since the LM step was not successful for even the optimized function (the subsample version of the true objective).
2. $S_K \leq 0$ but Eq. (13) is not satisfied: increase the sample set to K^+, λ remains unchanged.
3. $S_K \leq 0$ but Eq. (13) is satisfied: decrease λ (e.g. $\lambda \leftarrow \lambda/10$)

[4] Hoeffding's inequality is one of the main tools in statistical learning theory, but has seem limited use in computer vision so far (e.g. [40]).

[5] Note that these bounds may depend on the current iteration, hence a and b should be understood as $a^{(t)}$ and $b^{(t)}$.

Note that Hoeffding's inequality also holds for sample sets without replacement. This means that indices i_k can be unique and obtained by random shuffling the residuals at the beginning of the algorithm. Let π be a random permutation of $\{1, \ldots, N\}$. Then $\mathcal{S}_K$ is given by $\mathcal{S}_K = \{\pi(k) : k = 1, \ldots, K\}$. Thus, it is not necessary to draw batches at every iteration, which drastically reduces the variance of the iterates $\boldsymbol{\theta}^{(t)}$. Further, using slightly more general versions of Hoeffding's inequality allows residual specific upper and lower bounds $[a_i, b_i]$, which can be useful especially when residuals can be grouped (e.g. into groups for the data terms and for a regularizer).

4.2 Bounding the change of residuals

Lower bound a Observe that, both the l.h.s. and r.h.s. of the criterion (13) depend on the lower bound a. Due to the fact that $\mathbb{E}[Y_k] \leq \mathbb{E}[Z_k]$, the condition (13) is valid for any choices of $a < 0$. One fast option to search for a is to successively select the observed reductions Y_k as values for a, and test whether the condition (13) is satisfied for any of them.

Upper bound b While choosing the upper bound b used in (13) is generally hard, in practice it can be approximated using several options:

1. Each f_i has range $[0, \bar{f}]$. This is the case e.g. when all $\mathbf{r}_i$ are continuous and the domain for $\boldsymbol{\theta}$ is compact. It is also the case when f_i are robustified, i.e. $f_i(\boldsymbol{\theta}) = \psi(\|\mathbf{r}_i(\boldsymbol{\theta})\|)$, where $\psi : \mathbb{R}_{\geq 0} \to [0, 1]$ is a robust kernel (such as the Geman-McClure or the Welsch kernels). If the upper bound $\bar{f}$ for each f_i is known, then b in Eq. (13) is given by $b = \bar{y}$ (since the worst case increase of a term f_i is from 0 to $\bar{y}$).
2. Each f_i is Lipschitz continuous with constant L_f. In this case we have $|f_i(\boldsymbol{\theta}) - f_i(\boldsymbol{\theta}')| \leq L_f \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|$, in particular for $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)}$ and $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t+1)}$. This implies that $|f_i(\boldsymbol{\theta}^{(t+1)}) - f_i(\boldsymbol{\theta}^{(t)})| \leq L_f \|\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}\|$ for all i, and b in Eq. (13) is therefore given by $b = L_f \|\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}\|$. This computation of b can be extended straightforwardly if all f_i are Hölder continuous.

In our experiments, we test our algorithm on both robustified and non-robustified problems. In order to approximate L_f for non-robustified cases, we propose to use the maximum change in sampled residuals, i.e.,

$$L_f = \max_{i \in \mathcal{S}} \frac{|f_i(\boldsymbol{\theta}^{(t+1)}) - f_i(\boldsymbol{\theta}^{(t)})|}{\|\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}\|}. \tag{14}$$

4.3 Determining New Sample Sizes

At any iteration, when the condition (13) fails, the sample size K is increased to $K^+ > K$, and the algorithm continues with an extended subset $\mathcal{S}^+$ with $|\mathcal{S}^+| = K^+$. In this work, we approximate K^+ as follows: we use the estimate

$\hat{S}_{K^+} = K^+ S_K / K$ for S_{K^+} and choose K^+ such that the condition Eq. (13) is satisfied for our estimate $\hat{S}_{K^+}$. After simplification we obtain

$$K^+ = -\frac{K^2 (b-a)^2 \log \delta}{2 S_K^2 (1-\alpha)^2}. \tag{15}$$

If we introduce $\tilde{\delta} := \exp(2S_K^2/(K(b-a)^2))$ as the confidence level such that $P(\mathbb{E}[S_K] \geq 0) \leq \tilde{\delta}$ under the observed value S_K, then K^+ can be stated as $K^+ = K \frac{\log \delta}{\log \tilde{\delta}}$. If $K^+ > N$, then the new batch size is at most N. In summary, K^+ is given by

$$K^+ = \min \left\{ N, \left\lceil -\frac{K^2 (b-a)^2 \log \delta}{2 S_K^2 (1-\alpha)^2} \right\rceil \right\}. \tag{16}$$

4.4 Relaxing the Condition (13)

If T_S iterations of LM steps on subsampled residuals are applied, then the probability that *all* of these iterations led to a decrease of the full objective is given by $(1-\delta)^{T_S}$. Asking for all steps to be true descent steps can be very restrictive, and makes the condition (13) unnecessarily strict[6]. In practice one is interested that (i) most iterations (but not necessarily all) lead to a descent of the true objective, and that (ii) the true objective is reduced at the end of the algorithm (or for a number of iterations) with very high probability. Let t_0 and t be two iterations counters $t_0 < t$ such that the sample sets are constant, $\mathcal{S}^{(t_0)} = \ldots = \mathcal{S}^{(t)}$. Let $T_S = t - t_0 + 1$ be the number of successful LM iterations, that use the current sample set $\mathcal{S}^{(t)}$, and introduce the total observed reduction of the sampled cost after T (successful) iterations,

$$U_K^{(t_0,t)} := \sum\nolimits_{r=t_0}^{t} S_K^{(r)}, \tag{17}$$

and recall that $S_K = U_K^{(t_0,t_0)}$. Let the current iteration be a successful step (leading to a reduction of the sampled objective $f_{\mathcal{S}_K^{(t)}}$). With the introduction of U_k, following the same reasoning as introduced in Sec. 4.1, our relaxed criterion reads:

$$U_K^{(t_0,t)} \leq -\frac{1-\alpha}{b-a} \sqrt{\frac{-K \log \delta}{2}} \tag{18}$$

If the above criterion (with $\alpha \in (0,1)$) is not satisfied, then with probability $\eta \in [0,1)$ the step is temporarily accepted (and λ reduced). With probability $1-\eta$ the step is rejected and the sample size is increased. The rationale is that allowing further iterations with the current sample set may significantly reduce the objective value. If the condition (18) is never satisfied for the current sample set $\mathcal{S}^{(t_0)}$, then the expected number of "wasted" iterations is $1/(1-\eta)$ (using the properties of the geometric series).

[6] If we allow a "failure" probability η_0 for only increasing steps, then δ is given by $\delta = 1 - \sqrt[T_S]{(1-\eta_0)}$. E.g., setting $T_S = 100$ and $\eta_0 = 10^{-4}$ yields $\delta \approx 10^{-6}$.

Algorithm 1 Stochastic Levenberg-Marquardt

Require: Initial solution $\boldsymbol{\theta}^{(0)}$, initial batch size K_0, maximum iterations `max_iter`
Require: Confidence level $\delta \in (0,1)$, margin parameter $\alpha \in [0,1)$

Randomly shuffle the residuals $\{f_i\}$ and initialize $t \leftarrow 0$, $K \leftarrow K_0$
while $t <$ `max_iter` and a convergence criterion is not met **do**
  $\mathcal{S}^{(t)} \leftarrow \{1, \ldots, K\}$
  Compute $\mathbf{g}_{\mathcal{S}^{(t)}}$ and $\mathbf{H}_{\mathcal{S}^{(t)}}$

$$\mathbf{g}_{\mathcal{S}^{(t)}} \coloneqq \sum\nolimits_{i \in \mathcal{S}^{(t)}} \mathbf{J}_i^{(t)} \mathbf{r}_i^{(t)} \qquad \mathbf{H}_{\mathcal{S}^{(t)}} \coloneqq \sum\nolimits_{i \in \mathcal{S}^{(t)}} (\mathbf{J}_i^{(t)})^T \mathbf{J}_i^{(t)}. \tag{19}$$

  and solve

$$\Delta\boldsymbol{\theta}^{(t)} \leftarrow \left(\mathbf{H}_{\mathcal{S}^{(t)}} + \lambda \mathbf{I}\right)^{-1} \mathbf{g}_{\mathcal{S}^{(t)}} \qquad \boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} + \Delta\boldsymbol{\theta}^{(t)} \tag{20}$$

  Determine current lower and upper bounds a and b, and set

$$S_K \leftarrow \sum\nolimits_{i \in \mathcal{S}^{(t)}} \max\left\{a, \left(f_i(\boldsymbol{\theta}^{(t+1)}) - f_i(\boldsymbol{\theta}^{(t)})\right)\right\}. \tag{21}$$

  if $S_K \geq 0$ **then**
    $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)}$ and $\lambda \leftarrow 10\,\lambda$ ▷ Failure step
  else if S_K satisfies Eq. (13) **then**
    $\lambda \leftarrow \lambda/10$ ▷ Success step
  else
    $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)}$ and increase K using Eq. (16)
  end if
  $t \leftarrow t + 1$
end while
return $\boldsymbol{\theta}^{(t)}$

4.5 The complete algorithm

We illustrate the complete method in Alg. 1, which essentially follows a standard implementation of the Levenberg-Marquardt method. One noteworthy difference is that the implementation distinguishes between three scenarios depending on the reduction gain S_K (failure step, success step and insufficient step). For clarity we describe the basic (non-relaxed) variant of the method, and refer to the supplementary material for the implementation based on the relaxed test (Eq. (18) and the details of estimating the lower bound a. In the experiments we refer to our algorithm using the acronym *ProBLM* (**Pro**gressive **B**atching **LM**).

4.6 Convergence

When `max_iter`$\rightarrow \infty$, then the convergence properties of the algorithm are the same as for the regular Levenberg-Marquardt method: if a sample set $\mathcal{S}^{(t)}$ remains constant for a number of iterations, the method eventually approaches a stationary point of $f_{\mathcal{S}^{(t)}}$ leading to diminishing reductions S_K. Consequently,

Eq. (13) will not hold after a finite number of iterations, and the batch size strictly increases until $K^+ = N$ is reached.

5 Experiments

We choose dense image alignment (with homography model and photometric errors), and essential matrix estimation (with geometric errors) to evaluate the performance our proposed algorithm. Experiments for bundle adjustment can be found in the supplementary. The image pairs used throughout our experiments are obtained from a variety of publicly available datasets, including the ETH3D[7], EPFL Multi-view stereo[8] and AdelaideMRF[9] dataset [41]. In this section, we focus on presenting representative results that highlight the performance of our approach. More detailed results and studies of parameters are provided in the supplementary material. Two types of problems are tested in our experiments:

- Standard NLLS (Problem (1)): To test this type of problem, we perform dense homography estimation with photometric errors, and essential matrix refinement using sparse key points (outliers are assumed to be rejected by a pre-processing step, e.g. using RANSAC [13]).
- Problems with robustified residuals: We also investigate the performance of our approach on model fitting problems with robust kernels (Problem (2)). The essential matrix estimation on a sparse set of putative correspondences (containing outliers) is performed, where the outliers are directly discarded during the optimization process by applying a robust kernel ψ to the residuals (in contrast to the previous experiments where RANSAC is used to discard outliers). We choose ψ to be the smooth truncated least squares kernel,

$$\psi(r) = \tfrac{\tau^2}{4}\left(1 - \max\{0, 1 - r^2/\tau^2\}^2\right) \tag{22}$$

  where τ is the inlier threshold. In this case, the upper bound b on the residual changes that is used in Eq. (13) is $\frac{1}{4}\tau^2$.

The standard LM algorithm is used as the baseline to assess the performance of our proposed approach. In addition, we also compared our method against L-BFGS. All algorithms are implemented in C++ and executed on an Ubuntu workstation with an AMD Ryzen 2950X CPU and 64GB RAM. We employ the open-source OpenCV library[10] for pre-processing tasks such as SIFT feature extraction and robust fitting with RANSAC. We set δ to 0.1 and α to 0.9 in all experiments. The initial sample size (K_0) is set to $0.1N$ (N is the number of total measurements). All the experiments use the relaxed version as shown in Eq. (18), where the parameter η is set to 0.5. A comparison between Eq. (13) and its relaxed version is provided in the supplementary material.

[7] `https://www.eth3d.net/datasets`
[8] `https://www.epfl.ch/labs/cvlab/data/data-strechamvs/`
[9] `https://tinyurl.com/y9u7zmqg`
[10] `https://github.com/opencv/opencv`

5.1 Dense Image Alignment with Photometric Errors

This problem is chosen to demonstrate the efficiency of our proposed method as it often requires optimizing over a very large number of residuals (the number of pixels in the source image). In particular, given two images I_1 and I_2, the task is to estimate the parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ that minimize the photometric error,

$$\min_{\boldsymbol{\theta}} \sum\nolimits_{\mathbf{x} \in I_1} \|I_1(\mathbf{x}) - I_2(\pi(\mathbf{x}, \boldsymbol{\theta}))\|^2, \tag{23}$$

where $\mathbf{x}$ represents the pixel coordinates, and $\pi(\mathbf{x}, \boldsymbol{\theta})$ is the transform operation of a pixel $\mathbf{x}$ w.r.t. the parameters $\boldsymbol{\theta}$. In this experiment π is chosen to be the homography transformation, thus $\boldsymbol{\theta} \in \mathbb{R}^8$ (as the last element of the homography matrix can be set to 1). When linearizing the residual we utilize the combined forward and backward compositional approach [42] (which averages the gradient contribution from I_1 and I_2), since this is more stable in practice and therefore a more realistic scenario.

We select six image pairs from the datasets (introduced above) to test our method (results for more image pairs can be found in the supplementary materials). Fig. 1 shows the optimization progress for the chosen image pairs, where we plot the evolution of the objectives vs. the run times for our method and conventional LM. We also compare the results against L-BFGS, where it can clearly be observed that L-BFGS performs poorly for this particular problem (note that for image pairs where L-BFGS does not provide reasonable solutions, we remove their results from the plots).

As can be observed from Fig. 1, ProBLM achieves much faster convergence rates compared to LM. Moreover, Fig. 1 also empirically shows that our proposed method always converges to the same solutions as LM, thanks to our efficient progressive batching mechanism. Due to the non-linearity of the underlying problem this is somewhat surprising, since the methods will follow different trajectories in parameter space.

5.2 Essential Matrix Estimation with Squared Residuals

High-resolution image pairs from the *"facade"* sequence extracted from the ETH3D dataset are used in this experiment. For each image, we extract SIFT key points, and use nearest neighbor search to get approximately 5000 putative correspondences per image pair. The key points are normalized using the corresponding intrinsic matrices provided with the dataset. To obtain an outlier-free correspondence set, we run 100 RANSAC iterations on the putative matches to obtain around 2000 inliers per image pair, which are then fed into the non-linear least squares solvers for refinement. The objective of interest is the total Sampson error induced by all residuals, and we use the parameterisation of [14].

We first evaluate the performance of the algorithms on a single pair of images with different random starting points. Fig. 2a shows the objectives versus run time for a single pair of image on 20 runs, where at the beginning of each run, a random essential matrix is generated and used as starting solution for all

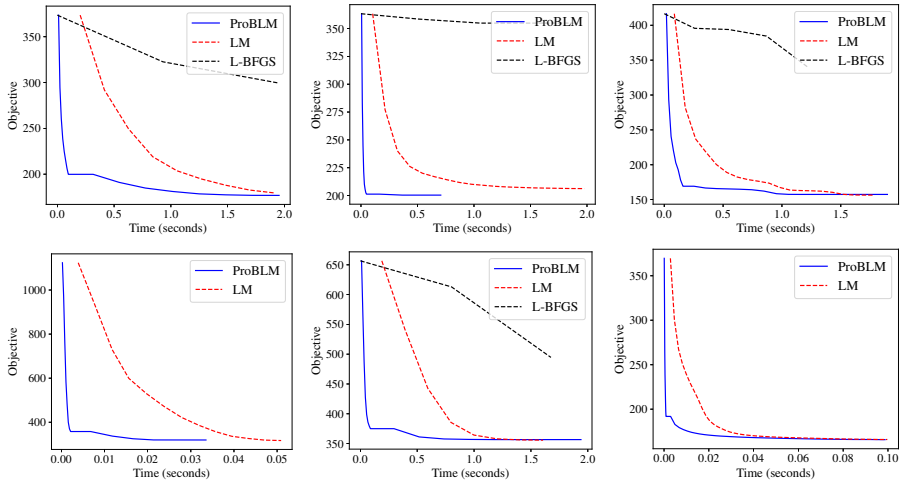


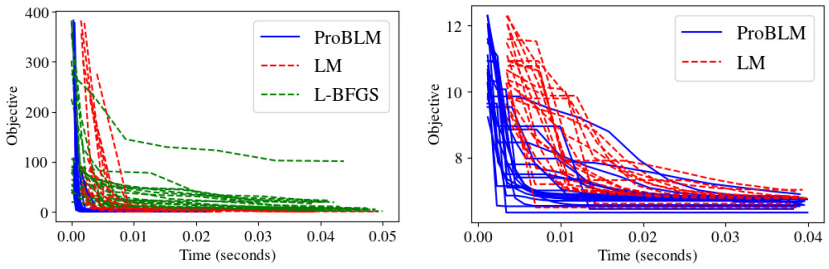
Fig. 1: Plots of objective vs. time for our method in comparison with LM and L-BFGS on dense image alignment. The image pairs used are (from left to right, top to bottom): Head, Hartley Building, Union House, Old Classics Wing, Johnson, and Napier Building.

methods. Similar to the case of dense image alignment shown in Fig. 1, ProBLM demonstrates superior performance throughout all the runs.

The experiment is repeated for 50 different pairs of images. For each pair, we execute 100 different runs and record their progresses within a run time budget of $10ms$. The results are summarized in Fig. 3a, where we use performance profiles [43] to visualize the overall performance. For each image pair, we record the minimum objective f^* obtained across 100 runs, then measure the percentage of runs (denoted by ρ) that achieves the cost of $\leq \tau f^*$ ($\tau \geq 1$) at termination. Fig. 3a shows the results. Observe that within a time budget of $10ms$, a large fraction of ProBLM runs achieve the best solutions, while most LM runs take much longer time to converge. This shows that our method is of great interest for real-time applications.

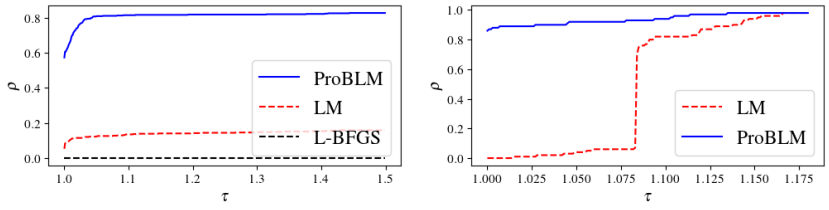
5.3 Robust Essential Matrix Fitting

As introduced earlier, our method is directly applicable to model fitting problems with robust kernels. To demonstrate this, we repeat the essential matrix fitting problem as discussed in the previous section, but we use the set of 5000 putative correspondences as input. To enforce robustness, we apply the smooth truncated least squares kernel shown in Eq. (22). Graduated Non-convexity [44] with 5 graduated levels is employed as the optimization framework. At each level (outer loop), our method is used to replace LM and the problem is optimized until convergence before switching to the next level. We compare this traditional approach where LM is used in the nested loop. Fig. 2b and 3b show



(a) Essential Matrix Estimation (b) Robust Essential Matrix

Fig. 2: Objectives vs run-time for 20 runs with random initializations for non-robust (left) and robust essential matrix essential matrix estimation (right).



(a) Essential Matrix Estimation (b) Robust Essential Matrix

Fig. 3: Performance profiles for (left): essential matrix estimation with run time budget set to 10ms, and (right): robust essential matrix estimation with run time budget set to 200 ms.

the evolution and performance profile (with the time budget of $200ms$) for this experiment. Similar to the case of clean data, our proposed method outperforms traditional LM by a large margin. A comparison with RANSAC can be found in the supplementary material, where we demonstrate that by applying ProBLM, one achieves comparable solutions to RANSAC within the same amount of run time, which further strengthens the applicability of our method for a wide range of vision problems.

6 Conclusion

We propose to accelerate the Levenberg-Marquardt method by utilizing sub-sampled estimates for the gradient and approximate Hessian information, and by dynamically adjusting the sample size depending on the current progress. Our proposed method has a straightforward convergence guarantee, and we demonstrate superior performance in model fitting tasks relevant in computer vision.

One topic for future research is to investigate in advanced algorithms addressing large-scale and robustified non-linear least-squares problems in order to improve the run-time performance and the quality of the returned solution.

References

1. Madsen, K., Nielsen, N., Tingleff, O.: Methods for non-linear least squares problems. Technical report, Technical University of Denmark (2004)
2. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge university press (2003)
3. Baker, S., Matthews, I.: Lucas-Kanade 20 years on: A unifying framework. International journal of computer vision **56** (2004) 221–255
4. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle adjustmenta modern synthesis. In: International workshop on vision algorithms, Springer (1999) 298–372
5. Nocedal, J., Wright, S.: Numerical optimization. Springer Science & Business Media (2006)
6. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. Quarterly of applied mathematics **2** (1944) 164–168
7. Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. Journal of the society for Industrial and Applied Mathematics **11** (1963) 431–441
8. Golub, G.H., Pereyra, V.: The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. SIAM Journal on Numerical Analysis **10** (1973) 413–432
9. Okatani, T., Deguchi, K.: On the wiberg algorithm for matrix factorization in the presence of missing components. International Journal of Computer Vision **72** (2007) 329–337
10. Hong, J.H., Zach, C., Fitzgibbon, A.: Revisiting the variable projection method for separable nonlinear least squares problems. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. (2017) 127–135
11. Jurie, F., Dhome, M.: Hyperplane approximation for template matching. IEEE Transactions on Pattern Analysis and Machine Intelligence **24** (2002) 996–1000
12. Agarwal, S., Snavely, N., Seitz, S.M., Szeliski, R.: Bundle adjustment in the large. In: European conference on computer vision, Springer (2010) 29–42
13. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24** (1981) 381–395
14. Rosten, E., Reitmayr, G., Drummond, T.: Improved ransac performance using simple, iterative minimal-set solvers. Technical report (2010)
15. Davis, T.A., Gilbert, J.R., Larimore, S.I., Ng, E.G.: Algorithm 836: Colamd, a column approximate minimum degree ordering algorithm. ACM Transactions on Mathematical Software (TOMS) **30** (2004) 377–380
16. Kalman, R.E.: A new approach to linear filtering and prediction problems. Transactions of the ASME–Journal of Basic Engineering **82** (1960) 35–45
17. Pearl, J.: Reverend Bayes on inference engines: A distributed hierarchical approach. In: Proceedings of the Second AAAI Conference on Artificial Intelligence. AAAI82, AAAI Press (1982) 133136
18. Drummond, T., Cipolla, R.: Real-time tracking of highly articulated structures in the presence of noisy measurements. In: Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001. Volume 2. (2001) 315–320 vol.2
19. Robbins, H., Monro, S.: A stochastic approximation method. The annals of mathematical statistics (1951) 400–407

20. Kiefer, J., Wolfowitz, J., et al.: Stochastic estimation of the maximum of a regression function. The Annals of Mathematical Statistics **23** (1952) 462–466
21. Wilson, D.R., Martinez, T.R.: The general inefficiency of batch training for gradient descent learning. Neural networks **16** (2003) 1429–1451
22. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. arXiv preprint arXiv:1609.04836 (2016)
23. Byrd, R.H., Hansen, S.L., Nocedal, J., Singer, Y.: A stochastic quasi-newton method for large-scale optimization. SIAM Journal on Optimization **26** (2016) 1008–1031
24. Bollapragada, R., Nocedal, J., Mudigere, D., Shi, H.J., Tang, P.T.P.: A progressive batching l-bfgs method for machine learning. In: International Conference on Machine Learning. (2018) 620–629
25. Zhao, R., Haskell, W.B., Tan, V.Y.: Stochastic l-bfgs: Improved convergence rates and practical acceleration strategies. IEEE Transactions on Signal Processing **66** (2018) 1155–1169
26. Curtis, F.E., Shi, R.: A fully stochastic second-order trust region method. arXiv preprint arXiv:1911.06920 (2019)
27. Nocedal, J.: Updating quasi-newton matrices with limited storage. Mathematics of computation **35** (1980) 773–782
28. Liu, D.C., Nocedal, J.: On the limited memory bfgs method for large scale optimization. Mathematical programming **45** (1989) 503–528
29. Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. Siam Review **60** (2018) 223–311
30. Tran-Dinh, Q., Pham, N.H., Nguyen, L.M.: Stochastic gauss-newton algorithms for nonconvex compositional optimization. arXiv preprint arXiv:2002.07290 (2020)
31. Byrd, R.H., Chin, G.M., Nocedal, J., Wu, Y.: Sample size selection in optimization methods for machine learning. Mathematical programming **134** (2012) 127–155
32. Bollapragada, R., Byrd, R., Nocedal, J.: Adaptive sampling strategies for stochastic optimization. SIAM Journal on Optimization **28** (2018) 3312–3343
33. Mohr, R., Stein, O.: An adaptive sample size trust-region method for finite-sum minimization. arXiv preprint arXiv:1910.03294 (2019)
34. Agarwal, N., Bullins, B., Hazan, E.: Second-order stochastic optimization for machine learning in linear time. The Journal of Machine Learning Research **18** (2017) 4148–4187
35. Pilanci, M., Wainwright, M.J.: Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. SIAM Journal on Optimization **27** (2017) 205–245
36. Roosta-Khorasani, F., Mahoney, M.W.: Sub-sampled newton methods. Mathematical Programming **174** (2019) 293–326
37. Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. SIAM Journal on scientific computing **16** (1995) 1190–1208
38. Holland, P.W., Welsch, R.E.: Robust regression using iteratively reweighted least-squares. Communications in Statistics-theory and Methods **6** (1977) 813–827
39. Hoeffding, W.: Probability inequalities for sums of bounded random variables. In: The Collected Works of Wassily Hoeffding. Springer (1994) 409–426
40. Cohen, A., C.Zach: The likelihood-ratio test and efficient robust estimation. In: IEEE International Conference on Computer Vision (ICCV). (2015)

41. Wong, H.S., Chin, T.J., Yu, J., Suter, D.: Dynamic and hierarchical multi-structure geometric model fitting. In: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE (2011) 1044–1051
42. Malis, E.: Improving vision-based control using efficient second-order minimization techniques. In: IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004. Volume 2., IEEE (2004) 1843–1848
43. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical programming **91** (2002) 201–213
44. Zach, C., Bourmaud, G.: Descending, lifting or smoothing: Secrets of robust cost optimization. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 547–562