An incremental trust-region method for Robust online sparse least-squares estimation

Citation: Rosen, David M., Michael Kaess, and John J. Leonard. "An incremental trust-region method for Robust online sparse least-squares estimation." Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA) (2012): 1262–1269.

As Published: http://dx.doi.org/10.1109/ICRA.2012.6224646

Publisher: Institute of Electrical and Electronics Engineers (IEEE)

Persistent URL: http://hdl.handle.net/1721.1/78897

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

An Incremental Trust-Region Method for Robust Online Sparse Least-Squares Estimation

David M. Rosen, Michael Kaess, and John J. Leonard

***Abstract*—Many online inference problems in computer vision and robotics are characterized by probability distributions whose factor graph representations are sparse and whose factors are all Gaussian functions of error residuals. Under these conditions, maximum likelihood estimation corresponds to solving a sequence of sparse least-squares minimization problems in which additional summands are added to the objective function over time. In this paper we present Robust Incremental least-Squares Estimation (RISE), an incrementalized version of the Powell's Dog-Leg trust-region method suitable for use in online sparse least-squares minimization. As a trust-region method, Powell's Dog-Leg enjoys excellent global convergence properties, and is known to be considerably faster than both Gauss-Newton and Levenberg-Marquardt when applied to sparse least-squares problems. Consequently, RISE maintains the speed of current state-of-the-art incremental sparse least-squares methods while providing superior robustness to objective function nonlinearities.**

I. Introduction

Many online inference problems in computer vision and robotics are characterized by probability distributions whose factor graph representations are sparse and whose factors are all Gaussian functions of error residuals; for example, both bundle adjustment [6] and the smoothing formulation of simultaneous localization and mapping (SLAM) [19], [20] belong to this class (Fig. 1). Under these conditions, maximum likelihood estimation corresponds to solving a sequence of sparse least-squares minimization problems in which additional summands are added to the objective function over time.

In practice, these problems are often solved by computing each estimate in the sequence as the solution of an independent minimization problem using standard sparse least-squares techniques (usually Levenberg-Marquardt). While this approach is general and produces good results, it is computationally expensive, and does not exploit the sequential nature of the underlying inference problem; this limits its utility in real-time online applications, where speed is critical.

More sophisticated solutions exploit sequentiality by using recursive estimation. For example, in the context of SLAM, Dellaert and Kaess *et al.* developed *incremental smoothing and mapping* (iSAM) [10], [9], which uses the information gained from new data to produce a direct *update* to the previous estimate, rather than computing a new estimate from scratch. This incremental approach enables iSAM to achieve computational speeds unmatched by iterated batch techniques. However, iSAM internally uses the Gauss-Newton method to perform the least-squares minimization. While this method generally performs well when initialized with an estimate that is close to a local minimum of the objective function, it can exhibit poor (even divergent) behavior when applied to objective functions with significant nonlinearity. Overcoming this brittleness in the face of nonlinearity is essential for the development of robust general-purpose online inference methods.

The authors are with the Massachusetts Institute of Technology, Cambridge, MA 02139, USA {dmrosen,kaess,jleonard}@mit.edu

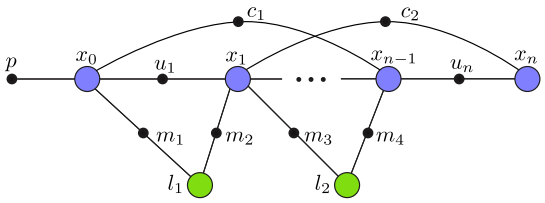


Fig. 1. Factor graph formulation of the SLAM problem, where variable nodes are shown as large circles, and factor nodes (measurements) as small solid circles. This example combines the pose-graph and the landmark-based SLAM formulations. The factors shown are odometry measurements u, a prior p, loop closing constraints c and landmark measurements m.

In this paper, we adopt the Powell's Dog-Leg optimization algorithm as the basis for sparse least-squares estimation. Powell's Dog-Leg is known to perform significantly faster than Levenberg-Marquardt in sparse least-squares minimization while achieving comparable accuracy [14]. Furthermore, by exploiting pre-existing functionality provided by the iSAM framework, it is possible to produce a fully incrementalized version of Powell's Dog-Leg suitable for use in online least-squares estimation. This incrementalized algorithm, which we refer to as Robust Incremental least-Squares Estimation (RISE), maintains the speed of current state-of-the-art incremental sparse least-squares methods like iSAM while providing superior robustness to objective function nonlinearities.

II. Inference in Gaussian factor graphs

A *factor graph* is a bipartite graph $G = (\mathcal{F}, \Theta, \mathcal{E})$ with two node types: *factor nodes* $f_i \in \mathcal{F}$ (each representing a real-valued function) and *variable nodes* $\theta_j \in \Theta$ (each representing an argument to one or more of the functions in $\mathcal{F}$). Every real-valued function $f(\Theta)$ has a corresponding factor graph G encoding its factorization as

$$\begin{aligned} f(\Theta) &= \prod_i f_i(\Theta_i), \\ \Theta_i &= \{\theta_j \in \Theta \mid (f_i, \theta_j) \in \mathcal{E}\}. \end{aligned} \tag{1}$$

If the function f appearing in equation (1) is a probability distribution, then maximum likelihood estimation corresponds

to finding the variable assignment Θ^* that maximizes (1):

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} f(\Theta). \tag{2}$$

Suppose now that each of the factors on the right-hand side of (1) is a Gaussian function of an error residual:

$$f_i(\Theta_i) \propto \exp\left(-\frac{1}{2}\left\|h_i(\Theta_i) - z_i\right\|^2_{\Sigma_i}\right), \tag{3}$$

where $h_i(\Theta_i)$ is a measurement function, z_i is a measurement, and $\|e\|^2_\Sigma = e^T\Sigma^{-1}e$ is the squared Mahalanobis distance with covariance matrix Σ. Taking the negative logarithm of the factored objective function (1) corresponding to the Gaussian model (3) shows that the maximizer Θ^* in (2) is given by:

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \frac{1}{2}\sum_i \left\|h_i(\Theta_i) - z_i\right\|^2_{\Sigma_i}. \tag{4}$$

Equation (4) shows that under the assumption of Gaussian factors (3), maximum likelihood estimation over factor graphs is equivalent to an instance of the general nonlinear least-squares minimization problem:

$$\begin{aligned} x^* &= \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} S(x), \\ S(x) &= \sum_{i=1}^{m} r_i(x)^2 = \|r(x)\|^2 \end{aligned} \tag{5}$$

for $r\colon \mathbb{R}^n \to \mathbb{R}^m$.

Note that each factor in (1) gives rise to a summand in the least-squares minimization (4). Consequently, performing *online* inference, in which new measurements become available (equivalently, in which new factor nodes are added to G) over time, corresponds to solving a *sequence* of minimization problems of the form (5) in which new summands are added to the objective function $S(x)$ over time.

III. Review of iSAM

The incremental smoothing and mapping (iSAM) algorithm [10], [9] is a computationally efficient method for solving a sequence of minimization problems of the form (5) in which new summands are added to the objective function over time. iSAM achieves this efficient computation by directly exploiting the sequential nature of the problem: it obtains each successive solution in the sequence of minimization problems by *directly incrementally updating* the previous solution, rather than computing a new solution from scratch (an expensive batch operation). Internally, iSAM uses an incrementalized version of the Gauss-Newton method to perform the minimization. In this section, we review the general Gauss-Newton method and its incremental implementation in iSAM.

A. The Gauss-Newton method

The Gauss-Newton algorithm [4], [15] is an iterative numerical technique for estimating the minimizer x^* of a problem of the form (5) for $m \geq n$. Given an estimate $x^{(i)}$ for the minimum, the function r is locally approximated by its linearization $L^{(i)}(h)$ about $x^{(i)}$:

$$L^{(i)}(h) = J\left(x^{(i)}\right)h + r\left(x^{(i)}\right), \tag{6}$$

where

$$J\left(x^{(i)}\right) = \left.\frac{\partial r}{\partial x}\right|_{x=x^{(i)}} \in \mathbb{R}^{m\times n}, \tag{7}$$

and a revised estimate

$$x^{(i+1)} = x^{(i)} + h^{(i)}_{gn} \qquad i \geq 0 \tag{8}$$

is chosen so that the estimated value for $S\left(x^{(i+1)}\right)$ obtained by using the approximation (6) in place of r on the right-hand side of (5) is minimized:

$$h^{(i)}_{gn} = \underset{h\in\mathbb{R}^n}{\operatorname{argmin}} \left\|L^{(i)}(h)\right\|^2. \tag{9}$$

Provided that the Jacobian $J\left(x^{(i)}\right)$ is full-rank, the *Gauss-Newton step* $h^{(i)}_{gn}$ defined in (9) can be found as the unique solution of

$$R^{(i)}h^{(i)}_{gn} = d^{(i)}, \tag{10}$$

where

$$Q^{(i)}\begin{pmatrix} R^{(i)} \\ 0 \end{pmatrix} = J\left(x^{(i)}\right) \tag{11}$$

is the QR decomposition [5] of the Jacobian $J\left(x^{(i)}\right)$ and

$$\begin{pmatrix} d^{(i)} \\ e^{(i)} \end{pmatrix} = -\left(Q^{(i)}\right)^T \cdot r\left(x^{(i)}\right) \tag{12}$$

for $d^{(i)} \in \mathbb{R}^n$ and $e^{(i)} \in \mathbb{R}^{m-n}$. Furthermore, in that case, since $R^{(i)}$ is upper-triangular, equation (10) can be solved efficiently via back-substitution.

The complete Gauss-Newton method consists of iteratively applying equations (7), (11), (12), (10), and (8), in that order, until some stopping criterion is satisfied.

B. Incrementalizing the solution: iSAM

As shown at the end of Section II, the arrival of new data corresponds to *augmenting* the function $r = r_{old}\colon \mathbb{R}^n \to \mathbb{R}^m$ on the right-hand side of (5) to the function

$$\begin{aligned} \bar{r}&\colon \mathbb{R}^{n+n_{new}} \to \mathbb{R}^{m+m_{new}} \\ \bar{r}&\left(x_{old}, x_{new}\right) = \begin{pmatrix} r_{old}\left(x_{old}\right) \\ r_{new}\left(x_{old}, x_{new}\right) \end{pmatrix}, \end{aligned} \tag{13}$$

where here $r_{new}\colon \mathbb{R}^{n+n_{new}} \to \mathbb{R}^{m_{new}}$ is the set of new measurement functions (factor nodes in the graph G) and $x_{new} \in \mathbb{R}^{n_{new}}$ is the set of new system variables (variable nodes in G) introduced as a result of the new observations.

In the naïve application of the Gauss-Newton algorithm of Section III-A, the solution $x^* = (x^*_{old}, x^*_{new})$ for the augmented least-squares problem determined by (13) would be found by performing Gauss-Newton iterations (8) until convergence. However, in the sequential estimation problem we already have a good estimate $\hat{x}_{old}$ for the values of the old variables, obtained by solving the least-squares minimization problem (5) prior to the introduction of this new set of data. Furthermore, under the assumption of the Gaussian error models (3) the raw observations z_{new} in (4) are typically fairly close to their ideal (uncorrupted) values, which provides a good guess for initializing the new estimates $\hat{x}_{new}$. We thus

obtain a good initial guess $\hat{x} = (\hat{x}_{old}, \hat{x}_{new})$ for the Gauss-Newton algorithm.

Now, since we expect the initial estimate $\hat{x}$ to be close to the true minimizing value x^*, it is not necessary to iterate the Gauss-Newton algorithm until convergence after integration of *every* new observation; instead, a *single* Gauss-Newton step is computed and used to correct the initial estimate $\hat{x}$. The advantage to this approach is that it avoids having to compute the Jacobian $\bar{J}(\hat{x})$ for $\bar{r}$ from scratch (an expensive batch operation) each time new observations arrive; instead, iSAM efficiently obtains $\bar{J}(\hat{x})$ together with its QR decomposition by *updating* the Jacobian $J(\hat{x}_{old})$ and its QR decomposition.

Letting $x = (x_{old}, x_{new})$, the Jacobian $\bar{J}(x)$ for the newly augmented system (13) can be decomposed into block form as

$$\bar{J}(x) = \frac{\partial \bar{r}}{\partial x} = \begin{pmatrix} \frac{\partial r_{old}}{\partial x_{old}} & 0 \\ \frac{\partial r_{new}}{\partial x_{old}} & \frac{\partial r_{new}}{\partial x_{new}} \end{pmatrix} = \begin{pmatrix} J(x_{old}) & 0 \\ W & \end{pmatrix} \quad (14)$$

where

$$J(x_{old}) = \frac{\partial r_{old}}{\partial x_{old}} \in \mathbb{R}^{m \times n}$$

is the Jacobian of the previous function r_{old} and

$$W = \begin{pmatrix} \frac{\partial r_{new}}{\partial x_{old}} & \frac{\partial r_{new}}{\partial x_{new}} \end{pmatrix} = \frac{\partial r_{new}}{\partial x} \in \mathbb{R}^{m_{new} \times (n+n_{new})}. \quad (15)$$

Letting

$$J(\hat{x}_{old}) = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix}$$

be the QR decomposition for the old Jacobian, where $Q_1 \in \mathbb{R}^{m \times n}$ and $Q_2 \in \mathbb{R}^{m \times (m-n)}$, we have

$$\begin{aligned} \begin{pmatrix} Q_1 & 0 & Q_2 \\ 0 & I & 0 \end{pmatrix} \begin{pmatrix} R & 0 \\ W & \\ 0 & \end{pmatrix} &= \begin{pmatrix} Q_1 R & 0 \\ W & \end{pmatrix} \\ &= \begin{pmatrix} J(\hat{x}_{old}) & 0 \\ W & \end{pmatrix} \\ &= \bar{J}(\hat{x}), \end{aligned} \quad (16)$$

which gives a partial QR decomposition of $\bar{J}(\hat{x})$. This decomposition can be completed by using Givens rotations to zero out the remaining nonzero elements below the main diagonal. Let $G \in \mathbb{R}^{(n+m_{new}) \times (n+m_{new})}$ denote a matrix of Givens rotations (necessarily orthogonal) such that

$$G \begin{pmatrix} R & 0 \\ W & \end{pmatrix} = \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix} \quad (17)$$

where $\bar{R} \in \mathbb{R}^{(n+n_{new}) \times (n+n_{new})}$ is upper-triangular. Then defining

$$\bar{G} = \begin{pmatrix} G & 0 \\ 0 & I \end{pmatrix}, \qquad \bar{Q} = \begin{pmatrix} Q_1 & 0 & Q_2 \\ 0 & I & 0 \end{pmatrix} \bar{G}^T, \quad (18)$$

(so that $\bar{G}, \bar{Q} \in \mathbb{R}^{(m+m_{new}) \times (m+m_{new})}$ are also orthogonal), equations (16), (17) and (18) show that

$$\bar{J}(\hat{x}) = \bar{Q} \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix} \quad (19)$$

is a QR decomposition for the new Jacobian $\bar{J}(\hat{x})$. Now we can use (10) and (12) to compute the Gauss-Newton step $\bar{h}_{gn}$ for the augmented system:

$$\begin{aligned} -\bar{Q}^T \cdot \bar{r}(\hat{x}) &= -\begin{pmatrix} G & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} Q_1^T & 0 \\ 0 & I \\ Q_2^T & 0 \end{pmatrix} \begin{pmatrix} r_{old}(\hat{x}_{old}) \\ r_{new}(\hat{x}) \end{pmatrix} \\ &= \begin{pmatrix} G & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} -Q_1^T \cdot r_{old}(\hat{x}_{old}) \\ -r_{new}(\hat{x}) \\ -Q_2^T \cdot r_{old}(\hat{x}) \end{pmatrix} \\ &= \begin{pmatrix} G & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} d_{old} \\ -r_{new} \\ e_{old} \end{pmatrix} \\ &= \begin{pmatrix} \bar{d} \\ e_{new} \\ e_{old} \end{pmatrix} \\ &= \begin{pmatrix} \bar{d} \\ \bar{e} \end{pmatrix}, \end{aligned} \quad (20)$$

where

$$\begin{pmatrix} \bar{d} \\ e_{new} \end{pmatrix} = G \begin{pmatrix} d_{old} \\ -r_{new}(\hat{x}) \end{pmatrix} \quad (21)$$

for $\bar{d} \in \mathbb{R}^{n+n_{new}}$. The Gauss-Newton step $\bar{h}_{gn}$ used to correct the estimate $\hat{x}$ for the augmented system is then computed as the solution of

$$\bar{R} \bar{h}_{gn} = \bar{d}. \quad (22)$$

Equations (15), (17) and (21) show how to obtain the $\bar{R}$ factor of the QR decomposition of $\bar{J}(\hat{x})$ and the corresponding linear system (22) by *updating* the R factor and linear system (10) for the previous Jacobian $J(\hat{x}_{old})$ using Givens rotations. Since the updated factor $\bar{R}$ and the new right-hand side vector $\bar{d}$ are obtained by applying G directly to the augmented factor R in (17) and the augmented right-hand side vector d in (21), it is not necessary to explicitly form the orthogonal matrix $\bar{Q}$ in (18). Nor is it necessary to form the matrix G explicitly either; instead, the appropriate individual Givens rotations can be directly applied to the matrix in (17) and the right-hand side vector in (21). Furthermore, under the assumption that the factor graph G is sparse, the Jacobian $\bar{J}(\hat{x})$ will likewise be sparse, so only a small number of Givens rotations are needed in (17). Obtaining the linear system (22) by this method is thus a computationally efficient operation.

Finally, we observe that while relinearization is not needed after *every* new observation, the system should be periodically relinearized about its corrected estimate in order to perform a full Gauss-Newton iteration and obtain a better estimate of the local minimum (this is particularly true after observations which are likely to significantly alter the estimates of system variables). When relinearizing the system about the corrected estimate, the incremental updating method outlined above is no longer applicable; instead, the QR factorization of the Jacobian needs to be recomputed from scratch. While this is a slow batch operation, the factorization step can be combined with a variable reordering step [2] in order to reduce the fill-in in the resulting factor R, thereby maintaining sparsity and speeding up subsequent incremental computations.

IV. From Gauss-Newton to Powell's Dog-Leg

The incrementalized Gauss-Newton method outlined in Section III-B is computationally efficient, straightforward to implement, and enjoys rapid (up to quadratic [15, pg. 22]) convergence near the minimum. However, the assumption of the local linearity of r (which justifies the approximation $r(x^{(i)} + h) \approx L^{(i)}(h)$ used in (9)) means that the Gauss-Newton method can exhibit poor behavior when there is significant nonlinearity near the current linearization point. Indeed, convergence of the Gauss-Newton method is not guaranteed, not even locally(!), and it is not difficult to construct simple (even quadratic!) examples of r where the sequence of iterates $\{x^{(i)}\}$ computed by (8) simply fails to converge at all [4, p. 113]. Thus, while Gauss-Newton may be acceptable for use in many cases, a robust general-purpose optimization method must be more tolerant of nonlinearities in the function r.

One possible alternative to Gauss-Newton is the use of steepest descent methods, which (as their name suggests) simply generate a step h_{sd} in the direction along which the objective function decreases most rapidly, i.e., in the direction of the negative gradient of the objective function:

$$h_{sd} = -\alpha \nabla S(x)$$

for some scalar $\alpha > 0$. These methods enjoy excellent global convergence properties [4], but their convergence is often slow since they are first-order, thus limiting their utility for online applications.

In this paper, we adopt the *Powell's Dog-Leg* algorithm [15], [16] as the method of choice for performing the sparse least-squares minimization (5). This algorithm combines the rapid end-stage convergence speed of the Gauss-Newton algorithm with the excellent global convergence properties [1], [17], [18] of steepest descent methods. Indeed, when applied to sparse least-squares minimization problems, Powell's Dog-Leg performs significantly faster than Levenberg-Marquardt (the current *de facto* nonlinear optimization method in the robotics and computer vision communities) while maintaining comparable levels of accuracy [14].

Internally, Powell's Dog-Leg operates by maintaining a *region of trust*, a ball of radius Δ centered on the current linearization point x within which the linearization (6) is considered to be a good approximation for r; this trust-region is used to guide an adaptive interpolation between Gauss-Newton and steepest descent steps. When computing a dog-leg step h_{dl}, Gauss-Newton updates are preferred (due to their superior speed), and are considered reliable so long as they fall within the trust region; otherwise, the more reliable steepest descent steps (or some linear interpolation between the two lying within the trust region) are used instead (Algorithm 1).

As the Powell's Dog-Leg algorithm proceeds, the radius of the trust region Δ is varied adaptively according to the *gain ratio*

$$\rho = \frac{S(x) - S\left(x + h_{dl}\right)}{L(0) - L\left(h_{dl}\right)}, \tag{23}$$

which compares the *actual* reduction in the objective function value obtained by taking the proposed dog-leg step h_{dl} with the *predicted* reduction in function value using the linear approximation (6). Values of the gain ratio close to 1 indicate that the approximation (6) is performing well near the current linearization point, so that the radius Δ of the trust region can be increased to allow larger steps (hence more rapid convergence), while values close to 0 indicate that the linearization is a poor approximation, and Δ should be reduced accordingly (Algorithm 2).

Algorithm 1 Computing the dog-leg step h_{dl}

procedure COMPUTE_DOG-LEG(h_{gn}, h_{sd}, Δ)
  if $\|h_{gn}\| \leq \Delta$ **then**
    $h_{dl} \leftarrow h_{gn}$
  else if $\|h_{sd}\| \geq \Delta$ **then**
    $h_{dl} \leftarrow \left(\frac{\Delta}{\|h_{sd}\|}\right) h_{sd}$
  else
    $h_{dl} \leftarrow h_{sd} + \beta\left(h_{gn} - h_{sd}\right)$, where β is chosen such that $\|h_{dl}\| = \Delta$
  end if
  return h_{dl}
end procedure

Algorithm 2 Updating the trust-region radius Δ

procedure UPDATE_TRUST_RADIUS(ρ, h_{dl}, Δ)
  if $\rho > .75$ **then**
    $\Delta \leftarrow \max\left\{\Delta, 3 \cdot \|h_{dl}\|\right\}$
  else if $\rho < .25$ **then**
    $\Delta \leftarrow \Delta/2$
  end if
  return Δ
end procedure

Combining the above strategies produces the Powell's Dog-Leg algorithm (Algorithm 3). The algorithm takes as its arguments the function r defining the objective function S as in (5), an initial estimate x_0 for the minimizer of S, and an initial estimate Δ_0 for the trust-region radius, together with four other arguments (ϵ_1, ϵ_2, ϵ_3, and k_{max}) that control the stopping criteria.

We point out for clarity that the vector g appearing in Algorithm 3 is a vector parallel to the objective function gradient $\nabla S(x)$, since

$$g = J(x)^T \cdot r(x) = \frac{1}{2} \nabla S(x), \tag{24}$$

where the second equality follows by differentiating equation (5). Likewise, the scalar α that controls the size of the steepest-descent step $h_{sd} = -\alpha g$ is (once again) obtained by substituting the linear approximation (6) into the right-hand side of (5), and then setting α to be the scaling factor

Algorithm 3 Powell's Dog-Leg

1: **procedure** POWELLS_DOG-LEG(r, x_0, Δ_0, ϵ_1, ϵ_2, ϵ_3, k_{max})
2: $k \leftarrow 0$, $x \leftarrow x_0$, $\Delta \leftarrow \Delta_0$, $g \leftarrow J(x_0)^T \cdot r(x_0)$
3: $stop \leftarrow [(\|r(x)\|_\infty \leq \epsilon_3)$ **or** $(\|g\|_\infty \leq \epsilon_1)]$
4: **while** (**not** $stop$ **and** $(k < k_{max})$) **do**
5: Compute h_{gn} using (10).
6: Set $\alpha \leftarrow \|g\|^2 / \|J(x)g\|^2$.
7: Set $h_{sd} \leftarrow -\alpha g$.
8: Set $h_{dl} \leftarrow$COMPUTE_DOG-LEG(h_{gn}, h_{sd}, Δ).
9: **if** $\|h_{dl}\| \leq \epsilon_2 (\|x\| + \epsilon_2)$ **then**
10: $stop \leftarrow true$
11: **else**
12: Set $x_{proposed} \leftarrow (x + h_{dl})$.
13: Compute ρ using (23).
14: **if** $\rho > 0$ **then**
15: Set $x \leftarrow x_{proposed}$.
16: Set $g \leftarrow J(x)^T \cdot r(x)$.
17: $stop \leftarrow [(\|r(x)\|_\infty \leq \epsilon_3)$ **or** $(\|g\|_\infty \leq \epsilon_1)]$
18: **end if**
19: $\Delta \leftarrow$ UPDATE_TRUST_RADIUS(ρ, h_{dl}, Δ)
20: **end if**
21: $k \leftarrow (k + 1)$
22: **end while**
23: **return** x
24: **end procedure**

minimizing the resulting estimated objective function value:

$$\alpha = \underset{a \in \mathbb{R}}{\operatorname{argmin}} \|L(-ag)\|^2 = \frac{\|g\|^2}{\|J(x)g\|^2}. \tag{25}$$

Note that even though this method uses the approximation (6) to compute the steepest-descent step, it is still valid within the context of the trust-region approach, since any steepest-descent step h_{sd} that is proposed for use as the dog-leg step h_{dl} and leaves the region of trust is scaled down to lie within it (lines 4 and 5 in Algorithm 1).

V. RISE: INCREMENTALIZING POWELL'S DOG-LEG

In this section we present Robust Incremental least-Squares Estimation (RISE), which we obtain by deriving an incrementalized version of Powell's Dog-Leg (Algorithm 3) with the aid of iSAM.

Algorithm 1 shows how to compute the dog-leg step h_{dl} directly from the Gauss-Newton step h_{gn} and the steepest-descent step h_{sd}; as iSAM already implements an efficient incremental algorithm for computing h_{gn}, it remains only to compute h_{sd}. In turn, line 7 of Algorithm 3 shows that h_{sd} is computed in terms of the gradient direction vector g and scale factor α, which can be determined using equations (24) and (25), respectively. Thus, it suffices to determine efficient incrementalized versions of (24) and (25).

Letting $x = (x_{old}, x_{new})$ as before and substituting the block decompositions (13) and (14) into (24) produces

$$\begin{aligned} \bar{g} &= \bar{J}(\hat{x})^T \cdot \bar{r}(\hat{x}) \\ &= \begin{pmatrix} J(\hat{x}_{old})^T & W^T \\ 0 & \end{pmatrix} \cdot \begin{pmatrix} r_{old}(\hat{x}_{old}) \\ r_{new}(\hat{x}_{old}, \hat{x}_{new}) \end{pmatrix} \\ &= \begin{pmatrix} J(\hat{x}_{old})^T \cdot r_{old}(\hat{x}_{old}) \\ 0 \end{pmatrix} + W^T \cdot r_{new}(\hat{x}_{old}, \hat{x}_{new}). \end{aligned} \tag{26}$$

Comparing the right-hand side of (26) with (24), we recognize the product $J(\hat{x}_{old})^T \cdot r_{old}(\hat{x}_{old})$ as nothing more than $g = g_{old}$, the gradient direction vector of the original (i.e. unaugmented) system at the linearization point $\hat{x}_{old}$. Thus, (26) can be reduced to

$$\bar{g} = \begin{pmatrix} g \\ 0 \end{pmatrix} + W^T \cdot r_{new}(\hat{x}). \tag{27}$$

Since the matrix W is sparse and its row dimension is equal to the (small) number of new measurements *added* when the system is extended, equation (27) provides an efficient method for obtaining the new gradient direction vector $\bar{g}$ by *incrementally updating* the previous gradient direction vector g, as desired.

Furthermore, in addition to obtaining $\bar{g}$ from g using (27) in incremental update steps, we can also exploit computations already performed by iSAM to more efficiently directly batch-compute $\bar{g}$ during relinearization steps, when incremental updates cannot be performed. Substituting the QR decomposition (19) into (24), we compute:

$$\begin{aligned} \bar{g} &= \bar{J}(\hat{x})^T \cdot \bar{r}(\hat{x}) \\ &= \left(\bar{Q} \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix} \right)^T \cdot \bar{r}(\hat{x}) \\ &= \begin{pmatrix} \bar{R}^T & 0 \end{pmatrix} \cdot \bar{Q}^T \cdot \bar{r}(\hat{x}). \end{aligned} \tag{28}$$

Comparing the final line of (28) with (20) shows that

$$\bar{g} = \begin{pmatrix} \bar{R}^T & 0 \end{pmatrix} \cdot \left(- \begin{pmatrix} \bar{d} \\ \bar{e} \end{pmatrix} \right) = -\bar{R}^T \cdot \bar{d}. \tag{29}$$

The advantage of equation (29) versus equation (24) is that $\bar{R}$ is a sparse matrix of smaller dimension than $\bar{J}(\hat{x})$, so that the matrix-vector multiplication in (29) will be faster. Moreover, since iSAM already computes the factor $\bar{R}$ and the right-hand side vector $\bar{d}$, the factors on the right-hand side of (29) are available at no additional computational expense.

Having shown how to compute the vector $\bar{g}$, it remains only to determine the scaling factor α as in (25). The magnitude of $\bar{g}$ can be computed efficiently directly from $\bar{g}$ itself, which gives the numerator of (25). To compute the denominator $\|\bar{J}(\hat{x})\bar{g}\|^2$, we again exploit the fact that iSAM already maintains the $\bar{R}$ factor of the QR decomposition for $\bar{J}(\hat{x})$; for since $\bar{Q}$ is orthogonal, then

$$\left\|\bar{J}(\hat{x})\bar{g}\right\|^2 = \left\|\left(\bar{Q}\bar{R}\right)\bar{g}\right\|^2 = \left\|\bar{R}\bar{g}\right\|^2,$$

and equation (25) is therefore equivalent to

$$\alpha = \|\bar{g}\|^2 / \|\bar{R}\bar{g}\|^2. \tag{30}$$

Again, since $\bar{R}$ is sparse, the matrix-vector multiplication appearing in the denominator of (30) is efficient.

Equations (27), (29), and (30) enable the implementation of RISE, a fully incrementalized version of Powell's Dog-Leg that integrates directly into the existing iSAM framework (Algorithm 4).

Furthermore, although these equations were derived in the context of the original iSAM algorithm [10] for pedagogical clarity, they work just as well in the context of iSAM2 [9]. In that case, the use of the Bayes Tree [8] and fluid relinearization enables the system to be efficiently relinearized at *every* timestep by applying direct updates only to those (few) rows of the factor $\bar{R}$ that are modified when relinearization occurs. One obtains the corresponding RISE2 algorithm by replacing lines 4 to 13 (inclusive) of Algorithm 4 with a different incremental update procedure for $\bar{g}$: writing $\bar{R}$ on the right-hand side of equation (29) as a stack of (sparse) row vectors, and then using knowledge of how the rows of $\bar{R}$ and the elements of $\bar{d}$ have been modified in the current timestep, enables the computation of an efficient incremental update to $\bar{g}$.

Finally, we point out that RISE(2)'s efficiency and incrementality are a direct result of exploiting iSAM(2)'s pre-existing functionality for computing the matrix R. In addition to being a purely intellectually pleasing result, this also means that any other computations depending upon pre-existing iSAM functionality (for example, online covariance extraction for data association [7]) can proceed with RISE *without modification*.

VI. Experimental results

In this section, we examine the performance of five algorithms (the Powell's Dog-Leg, Gauss-Newton, and Levenberg-Marquardt batch methods and the RISE and iSAM incremental methods) operating on a class of toy 6DOF SLAM problems in order to illustrate the superior performance of the Powell's Dog-Leg-based methods versus current state-of-the-art techniques. (We do not include experimental results for iSAM2 and RISE2 in this paper, as at the time of writing improved release implementations of these algorithms are in development. However, we expect that a comparison of the performance of iSAM2 and RISE2 will be qualitatively similar to what is demonstrated here for iSAM and RISE.)

Our test set consists of 1000 randomly generated instances of the sphere2500 data set [9]. The samples were generated using the executable `generateSpheresICRA2012.cpp` included in the iSAM version 1.6 release (available through http://people.csail.mit.edu/kaess/isam/).

Since one of our motivations in developing an incremental numerical optimization algorithm suitable for use in general nonlinear least-squares minimization is to enable the use of robust estimators in SLAM applications, we also replace the quadratic cost function in (4) with the pseudo-Huber robust cost function [6, p. 619]:

$$C(\delta) = 2b^2 \left(\sqrt{1 + (\delta/b)^2} - 1 \right)$$

with parameter $b = .5$.

Algorithm 4 The RISE algorithm

1: **procedure** RISE
2: Initialization: $\hat{x}_{old}$, $\hat{x}_{estimate} \leftarrow x_0$, $\Delta \leftarrow \Delta_0$.
3: **while** ($\exists$ new data $(\hat{x}_{new}, r_{new})$) **do**
4: **if** (relinearization_step) **then**
5: Update linearization point: $\hat{x}_{old} \leftarrow \hat{x}_{estimate}$.
6: Construct Jacobian $\bar{J}(\hat{x}_{old}, \hat{x}_{new})$.
7: Perform complete QR decomposition on $\bar{J}(\hat{x})$, cache $\bar{R}$ factor and right-hand side vector $\bar{d}$ as in equations (11) and (12).
8: Set $\bar{g} \leftarrow -\bar{R}^T \cdot \bar{d}$.
9: **else**
10: Compute the partial Jacobian W as in (15).
11: Obtain and cache the new $\bar{R}$ factor and new right-hand side vector $\bar{d}$ by means of Givens rotations as in equations (17) and (21).
12: Set

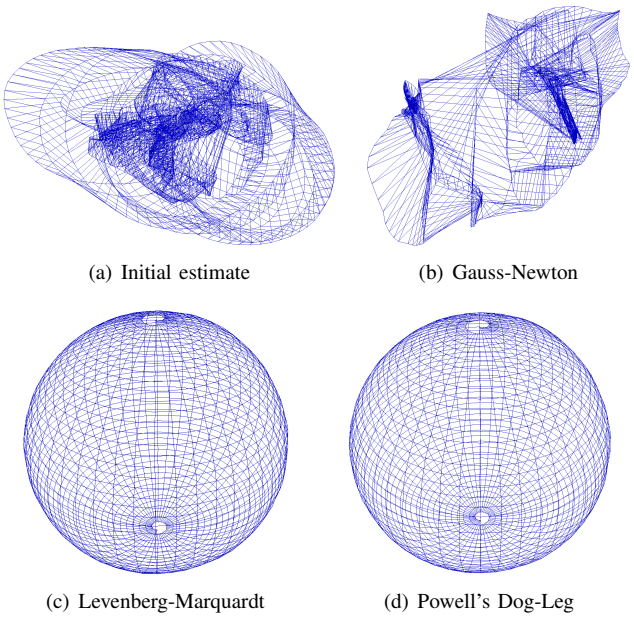
$$\bar{g} \leftarrow \begin{pmatrix} g \\ 0 \end{pmatrix} + W^T \cdot r_{new}(\hat{x}).$$

13: **end if**
14: Compute Gauss-Newton step h_{gn} using (22).
15: Set $\alpha \leftarrow \|\bar{g}\|^2 / \|\bar{R}\bar{g}\|^2$.
16: Set $h_{sd} \leftarrow -\alpha \bar{g}$.
17: Set $h_{dl} \leftarrow$COMPUTE_DOG-LEG(h_{gn}, h_{sd}, Δ).
18: Set $\hat{x}_{proposed} \leftarrow (\hat{x} + h_{dl})$.
19: Compute ρ using (23).
20: **if** $\rho > 0$ **then**
21: Update estimate: $\hat{x}_{estimate} \leftarrow \hat{x}_{proposed}$.
22: **else**
23: Retain current estimate: $\hat{x}_{estimate} \leftarrow \hat{x}$.
24: **end if**
25: Set $\Delta \leftarrow$UPDATE_TRUST_RADIUS(ρ, h_{dl}, Δ).
26: Update cached variables: $\hat{x}_{old} \leftarrow \hat{x}$, $r \leftarrow \bar{r}$, $g \leftarrow \bar{g}$, $R \leftarrow \bar{R}$, $d \leftarrow \bar{d}$.
27: **end while**
28: **return** $\hat{x}_{estimate}$
29: **end procedure**

All experiments were run on a desktop with Intel Xeon X5660 2.80 GHz processor. Each of the five algorithms used was implemented atop a common set of data structures, so any variation in algorithm performance is due *solely* to differences amongst the optimization methods themselves.

A. Batch methods

In this experiment, we compared the performance of the three batch methods to validate our hypothesis that Powell's Dog-Leg should be adopted as the standard for sparse least-squares minimization. Powell's Dog-Leg was initialized with $\Delta_0 = 1$, and the Levenberg-Marquardt algorithm (following the implementation given in Section A6.2 of [6] using additive modifications to the diagonal) was initialized with $\lambda_0 = 10^{-6}$ and scaling factor $\lambda = 10$ (the Gauss-Newton method takes



(a) Initial estimate (b) Gauss-Newton

(c) Levenberg-Marquardt (d) Powell's Dog-Leg

Fig. 2. A representative instance of the sphere2500 6DOF SLAM problem from the batch experiments. 2(a): The initial estimate for the solution (objective function value 1.221 E8). 2(b): The solution obtained by Gauss-Newton (3.494 E6). 2(c): The solution obtained by Levenberg-Marquardt (8.306 E3). 2(d): The solution obtained by Powell's Dog-Leg (8.309 E3). Note that the objective function value for each of these solutions is within $\pm 0.5\%$ of the median value for the corresponding method given in Table I.

no parameters). In all cases, the initial estimate for the robot path was obtained by integrating the simulated raw odometry measurements. All algorithms use the stopping criteria given in Algorithm 3, with $\epsilon_1 = \epsilon_2 = \epsilon_3 = .01$ and $k_{max} = 500$. The results of the experiment are summarized in Table I. A representative instance from the test set is shown in Fig. 2.

As expected, Powell's Dog-Leg and Levenberg-Marquardt achieved comparable levels of accuracy, significantly outperforming Gauss-Newton. The superior performance of these algorithms can be explained by their online adaptation: both Powell's Dog-Leg and Levenberg-Marquardt adaptively interpolate between the Gauss-Newton and steepest descent steps using the local behavior of the objective function as a guide, and both implement a look-ahead strategy in which the proposed step h is rejected if the objective function value at the proposed updated estimate $\hat{x}_{proposed}$ actually *increases* (cf. line 14 of Algorithm 3). In contrast, the Gauss-Newton method has no look-ahead strategy and no guard against overconfidence in the local linear approximation (6), which renders it prone to "overshooting" (i.e., taking too large a step), non-recoverably driving the estimate $\hat{x}$ away from the true local minimum.

In addition to its favorable accuracy, Powell's Dog-Leg is also the fastest algorithm of the three by an order of magnitude, both in terms of the number of iterations necessary to converge to a local minimum *and* the total elapsed computation time. The superior speed of Powell's Dog-Leg versus Gauss-Newton is explained by its adaptive step computations, which enable it to advance more directly toward local minima, whereas Gauss-Newton may take many false steps and overshoots before converging, or may not converge at all. The superior speed of Powell's Dog-Leg versus Levenberg-Marquardt has been studied previously [14]: it is due in part to the fact that Powell's Dog-Leg need only solve the normal equations *once* at each linearization point, whereas Levenberg-Marquardt must solve the modified normal equations (an expensive operation) whenever the linearization point is updated *or* the damping parameter λ is changed.

These results support our thesis (put forward in Section IV) that Powell's Dog-Leg should be adopted as the method of choice for general sparse nonlinear least-squares estimation.

B. Incremental methods

In this experiment, we compared the new RISE algorithm (Algorithm 4) with the original iSAM algorithm, a state-of-the-art incremental sparse least-squares method. RISE was initialized with $\Delta_0 = 1$, and both algorithms were set to relinearize every 100 steps. The results of the experiment are summarized in Table II. (Note that the statistics given for each method in the first and second rows of Table II are computed using only the set of problem instances for which that method did not terminate early, as explained below.)

As expected, RISE significantly outperformed the original iSAM in terms of accuracy. In over half of the problem instances, the solution computed by iSAM diverged so far from the true minimum that the numerically-computed Jacobian became rank-deficient, forcing the algorithm to terminate early (solving equation (10) requires that the Jacobian be full-rank). Even for those problem instances in which iSAM ran to completion (which are necessarily the instances that are the "easiest" to solve), Table II shows that the solutions computed using the incremental Gauss-Newton approach are considerably less accurate than those computed using the incremental Powell's Dog-Leg method. Indeed, RISE's performance on *all* of the problem instances was, on the average, significantly better than the original iSAM's performance on only the *easiest* instances.

While Table II shows that RISE is slightly slower than the original iSAM, this is to be expected: each iteration of the RISE algorithm must compute the Gauss-Newton step (the *output* of iSAM) as an intermediate result in the computation of the dog-leg step. Each of the steps in Algorithm 4 has asymptotic time-complexity at most $O(n)$ (assuming sparsity), which is the same as for iSAM, so we expect that RISE will suffer at worst a small constant-factor slowdown in speed versus iSAM. The results in Table II show that in practice this constant-factor slowdown has only a modest effect on RISE's overall execution speed (when the computational costs of manipulating the underlying data structures are also included): both iSAM and RISE are fast enough to run comfortably in real-time.

	Powell's Dog-Leg			Gauss-Newton			Levenberg-Marquardt		
	Mean	Median	Std. Dev.	Mean	Median	Std. Dev.	Mean	Median	Std. Dev.
Objective function value	8.285 E3	8.282 E3	71.40	4.544 E6	3.508 E6	4.443 E6	9.383 E3	8.326 E3	2.650 E3
Computation time (sec)	16.06	15.73	1.960	226.2	226.0	2.028	126.7	127.0	43.51
# Iterations	34.48	34	4.171	499.9	500	2.500	338.2	328	138.9
# Iteration limit interrupts	0			998			311		

TABLE I
SUMMARY OF RESULTS FOR BATCH METHODS

	RISE			iSAM		
	Mean	Median	Std. Dev.	Mean	Median	Std. Dev.
Objective function value	9.292 E3	9.180 E3	5.840 E2	6.904 E11	1.811 E4	1.242 E13
Computation time (sec)	50.21	50.18	0.13	42.97	42.95	0.13
# Early termination failures (rank-deficient Jacobian)	0 (0.0%)			586 (58.6%)		

TABLE II
SUMMARY OF RESULTS FOR INCREMENTAL METHODS

VII. CONCLUSION

In this paper we derived Robust Incremental least-Squares Estimation (RISE), an incrementalized version of the Powell's Dog-Leg trust-region method suitable for use in online sparse least-squares minimization. RISE maintains the speed of current state-of-the-art incremental sparse least-squares methods while providing superior robustness to objective function nonlinearities.

In addition to its utility as a general-purpose method for least-squares estimation, we expect that RISE will prove particularly advantageous in 6DOF lifelong mapping and visual SLAM tasks. In both of these applications, the large number of data association decisions that must be made make it likely that at least some of them will be decided incorrectly. Under the usual squared-error cost criterion, even a few erroneous data associations can severely degrade the quality of the resulting estimate. RISE's robustness to nonlinearity admits the use of robust cost functions in these applications, which should significantly attenuate the ill effects of erroneous data associations; indeed, this was our original motivation for developing the algorithm.

Finally, although RISE was developed under the assumption of Gaussian residual distributions, we believe that it is possible to generalize the algorithm to perform efficient online inference over arbitrary sparse factor graphs. Given the ubiquity of these models, this would be of considerable interest as a general-purpose online probabilistic inference method. We intend to explore this possibility in future research.

ACKNOWLEDGMENTS

This work was partially supported by Office of Naval Research (ONR) grants N00014-06-1-0043 and N00014-10-1-0936, and by Air Force Research Laboratory (AFRL) contract FA8650-11-C-7137. The views expressed in this work have not been endorsed by the sponsors.

REFERENCES

[1] R.G. Carter. On the global convergence of trust region algorithms using inexact gradient information. *SIAM Journal on Numerical Analysis*, 28(1):251–265, 1991.

[2] T. A. Davis, J. R. Gilbert, S. T. Larimore, and R. G. Ng. A column approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30:353–376, Sep 2004.

[3] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *Intl. J. of Robotics Research*, 25(12):1181–1203, Dec 2006.

[4] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 2nd edition, 1987.

[5] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[6] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[7] M. Kaess and F. Dellaert. Covariance recovery from a square root information matrix for data association. *Journal of Robotics and Autonomous Systems*, 57(12):1198–1210, Dec 2009.

[8] M. Kaess, V. Ila, R. Roberts, and F. Dellaert. The Bayes tree: An algorithmic foundation for probabilistic robot mapping. In *Intl. Workshop on the Algorithmic Foundations of Robotics, WAFR*, Singapore, Dec 2010.

[9] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J.J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[10] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. Robotics*, 24(6):1365–1378, Dec 2008.

[11] K. Konolige, G. Grisetti, R. Kummerle, W. Burgard, B. Limketkai, and R. Vincent. Sparse pose adjustment for 2d mapping. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 10/2010 2010.

[12] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[13] K. Levenberg. A method for the solution of certain nonlinear problems in least squares. *Quart. Appl. Math*, 2(2):164–168, 1944.

[14] M.I.A. Lourakis and A.A. Antonis. Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment? *Intl. Conf. on Computer Vision (ICCV)*, 2:1526–1531, 2005.

[15] K. Madsen, H.B. Nielsen, and O. Tingleff. *Methods for Non-Linear Least Squares Problems*. Informatics and Mathematical Modeling, Technical University of Denmark, 2nd edition, 2004.

[16] M.J.D. Powell. A new algorithm for unconstrained optimization. In J. Rosen, O. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*, pages 31–65. Academic Press, 1970.

[17] M.J.D. Powell. On the global convergence of trust region algorithms for unconstrained minimization. *Mathematical Programming*, 29(3):297–303, 1984.

[18] G.A. Shultz, R.B. Schnabel, and R.H. Byrd. A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties. *SIAM Journal on Numerical Analysis*, 22(1):47–67, 1985.

[19] S. Thrun. Robotic Mapping: A Survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[20] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2008.