$\mathcal{L}_1$ Factor Graph SLAM: going beyond the $\mathcal{L}_2$ norm

J.J Casafranca, L.M Paz, P. Piniés
Instituto de Investigación en Ingeniería de Aragón, Universidad de Zaragoza (Spain)
Email: {jjcasmar}@gmail.com,{linapaz,ppinies}@unizar.es

***Abstract*— In this paper we propose a novel solution based on the $\mathcal{L}_1$ norm as a *back-end* to pose graph SLAM. In contrast to other NLSQs approaches, we formulate an iterative algorithm inspired directly on the Factor Graph structure to solve for the linearized residual $\|\mathbf{Ax} - \mathbf{b}\|_1$. Although similar to the robust Huber norm, the $\mathcal{L}_1$ norm is much more effective in removing the effect of strong spurious data. Since our approach depends on the minimization of a non differentiable function, we provide the theoretical insights to solve for the $\mathcal{L}_1$ norm. Our optimization is based on a primal-dual formulation successfully applied for solving variational convex problems in computer vision. We show the effectiveness of the $\mathcal{L}_1$ norm to produce both a robust initial seed and a final optimized solution on challenging and well known datasets widely used in other state of the art works.**

I. Introduction

During the last recent years, graphical models and non linear optimization based algorithms have become the core of many state of the art SLAM and Bundle Adjustment approaches. In both contexts, the goal is to estimate a set of state variables representing the location of robot/sensor positions along a trajectory and the pose of geometric elements characterizing the environment structure. The most common way to formally define the problem is through a non linear least squares (NLSQs) formulation: given as input the set of relative measurements between state variables, the goal is to minimize an error cost function. A solution delivers the best state variable estimation for the cost function in a MAP sense. Graphical models aim to graphically interpret the stochastic relations between state variables and noisy observations. Lets consider the pose graph subproblem in which the set of state variables $\mathbf{x}$ represent the locations of the sensor/robot in the environment and $\mathbf{z}$ the measurements obtained from odometry readings or relative transformations calculated from scan matching. Figure 1 shows two common Graphical models used to represent a pose graph. On the left we have a Factor Graph model which is a bipartite graph with links between two types of nodes: first, nodes representing state variables $\mathbf{x}$ (blue circles); second, nodes (red squares) that explicitly codify the constraints $\mathbf{z}$ between the previous state variables. On the right of the figure we have a Gaussian Markov Random Field (GMRF) with a unique type of node representing the state variables whereas the links show the probabilistic relations that appear between the variables due to the indirect effect of the observations [1]. For typical pose graph problems in which the observations only link pairs of state variables both types of models turn out to produce very similar graph representations as can be observed in the figure.

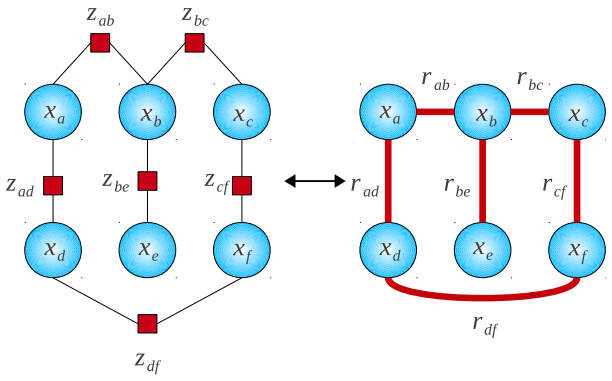


Fig. 1: On the left, a Factor Graph for a pose graph SLAM subproblem. A GMRF graph for the same problem structure (right)

Commonly, GMRF and Factor Graph solutions are based on minimizing an $\mathcal{L}_2$ norm function of the residuals $\|\mathbf{r}(\mathbf{x})\|_2^2$. A local quadratic approximation of the function is usually carried out by linearizing the residual errors $\mathbf{r} \approx \mathbf{Ax} - \mathbf{b}$. Iterative methods as gradient descend, conjugate gradient, Gauss-Newton or Levenberg Marquardt are used as minimizers [2], being Gauss-Newton and Levenberg Marquardt the most common algorithms.

In this paper we propose a novel solution based on the $\mathcal{L}_1$ norm as a *back-end* for solving a pose graph. We formulate an iterative algorithm based on the Factor Graph structure of the problem that solves in each step the $\mathcal{L}_1$ norm of the linearized residual $\|\mathbf{Ax} - \mathbf{b}\|_1$.

Our contributions are summarized as follows:

- We formulate a novel back-end for Factor Graph SLAM based on the $\mathcal{L}_1$ norm that is robust and effective in removing strong spurious data.
- We provide the theoretical insight for minimizing a non differentiable function like the $\mathcal{L}_1$ norm which is derived from well supported convex optimization theory. The optimization makes use of the primal-dual algorithm successfully applied in vision problems.
- Our solution method is easy to implement since requires simple calculations based on matrix-vector multiplications. We do not need to calculate the Hessian of the cost function or apply a reordering algorithm. In

This work was supported by the MICINN-FEDER project DPI2012-36070: HLSLAM - Life-long Active Spatial Cooperative Mapping and Understanding.

addition, we still exploit the sparseness of the Jacobian using the efficient SuiteSparse library for the matrix-vector multiplications.
- We propose a very simple and robust algorithm to produce a fast and good initial seed for the non linear $\mathcal{L}_1$ optimization.

II. Related Work

The most prominent works on pose graph optimization over the past few years [3], [4], [5], [6] highlight the importance of representing the whole estimation process as a Graphical model. These works show that the SLAM problem can be cast as a graph of relations between the involved state variables (the inferred data) and the observations (the evidence). The combination of an optimization algorithm and its graphical representation is a well known paradigm in the computer vision community in which it has been widely studied to solve the similar Bundle Adjustment problem [7]. In the robotics field, the work presented in [3] establishes a clear connection of GMRFs and Factor Graphs to the non linear SLAM problem. The solutions are based on matrix factorizations of the Hessian (GMRF) or the Jacobian (Factor Graph) of the cost function. For GMRF the Cholesky decomposition using CHOLMOD routines [8] is the common work-horse for factorizing the sparse Hessian. In [3] a QR factorization is used for the Jacobian while [9] introduces preconditioned conjugate gradient for large scale problems. The works in [4], [5] provide new contributions for incremental solutions. Similarly, [10], [11] find a similar way to explain the non linear optimization in terms of a Maximum Likelihood estimation. All this works forms the basis for the ***g2o*** popular library [12] used in pose graph optimization.

All previous works share in common the way of addressing the required optimization. They all deal with the $\mathcal{L}_2$ norm for the minimization of the residuals using a traditional NLQS formulation. It is well known that the $\mathcal{L}_2$ norm is not robust to the presence of spurious data. This fact has motivated the development of robust front-end stages in charge of discarding spurious links in the graph. Recently, new interesting efforts have been made to robustify the back-end stage either changing the cost function [13], or introducing switching variables to reduce the effect of spurious links [14]. These new approaches are also based on the $\mathcal{L}_2$ norm.

The idea of using a robust norm for the cost function has already been applied in many computer vision problems. For instance, the Huber norm is commonly used in Bundle Adjustment approaches to strength the optimization against the presence of spurious data [15]. In fact, the ***g2o*** library has as well included this norm as an option in its optimization process. Similarly, image processing applications make use of robust norms or prior regularizers such as the $\mathcal{L}_1$ norm or the Total Variation [16]. The main problem of these new norms and priors is their non-differentiable nature. Fortunately, the $\mathcal{L}_1$ norm belongs to the set of convex functions for which there is a well supported set of math tools from convex optimization theory and variational methods [17]. The advantage of convex problems over non convex ones is that a global optimum can be calculated with high accuracy in reasonable time, independently of the initial guess. In our work we implement as optimization kernel a primal-dual algorithm that has been successfully used to solve convex and non-differentiable optimization problems that appear in several image processing applications [18]. This algorithm is shown to converge in $O(1/k)$ where k is the number of iterations. In [19] it is shown that this rate of convergence is optimum for convex optimization problems with saddle point structure.

III. Minimizing $\|\mathbf{Ax}-\mathbf{b}\|_1$

The standard method to solve many robotic and computer vision problems is based on a NLSQ algorithm. The goal is to find a solution

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} F(\mathbf{x}) \tag{1}$$

by minimizing a function $F : \mathbf{R}^n \rightarrow \mathbf{R}$ of the form

$$F(\mathbf{x}) = \sum_{i,j} \|r(x_i, x_j, z_{ij})\|_2^2 = \|\mathbf{r}(\mathbf{x})\|_2^2$$

where $\mathbf{r}(\mathbf{x}) : \mathbf{R}^n \rightarrow \mathbf{R}^m$ is a nonlinear residual vector that measures the discrepancy of the relative relation between a pair of state variables $x_i, x_j \in \mathbf{x}$ and their gathered observation $z_{ij} \in \mathbf{z}$. A local approximation of the cost function is calculated by linearizing the set of residuals. The solution is then found iteratively by solving the $\mathcal{L}_2$ norm of the linearized residuals $\|\mathbf{Ax}-\mathbf{b}\|_2^2$.

The goal of this paper is to substitute the $\mathcal{L}_2$ norm by the more robust $\mathcal{L}_1$ norm in the optimization and iteratively solve for the linearized residual $\|\mathbf{Ax}-\mathbf{b}\|_1$. In order to tackle the non-differentiable nature of the $\mathcal{L}_1$ norm we will make use of the primal-dual optimization formulation.

A. Primal-Dual general problem

For completeness reasons this subsection gives a brief review of the Primal-Dual general problem for convex functions with saddle-point structure. A more detailed description can be found in [18]. The basic structure of the primal problem is given by

$$\min_{x \in \mathbf{X}} F(\mathbf{K}x) + G(x) \tag{2}$$

where $\mathbf{K} : \mathbf{X} \rightarrow \mathbf{Y}$ is a linear map between two finite-dimensional vector spaces $\mathbf{X}$ and $\mathbf{Y}$ equipped with an inner product $\langle \cdot\,, \cdot \rangle$ and norm $\| \cdot \| = \langle \cdot\,, \cdot \rangle^{\frac{1}{2}}$.

We transform the original primal problem in Eq. (2) to a saddle point problem by obtaining the Legendre-Fenchel transformation of the convex function F,

$$\min_{x \in \mathbf{X}} \max_{y \in \mathbf{Y}} \langle \mathbf{K}x, y \rangle + G(x) - F^*(y) \tag{3}$$

where $G\colon \mathbf{X} \rightarrow [0, +\infty)$ and $F^*\colon \mathbf{X} \rightarrow [0, +\infty)$ are proper, convex, lower semi-continuous (*l.s.c*) functions. Equation (3) is known as the primal-dual problem of (2) with dual variable

Algorithm 1 primal-dual

1: {Initialization of variables:}
2: $\tau, \sigma > 0, \theta \in [0,\ 1]$
3: $(x^0, y^0) \in \mathbf{X} \times \mathbf{Y}$
4: $\bar{x}^0 = x^0$
5: **while** $k \leq N$ **do**
6: {Update x^k,y^k,$\bar{x}^k$}
7: $y^{k+1} = (\mathbf{I} + \sigma \partial F^*)^{-1}(y^k + \sigma \mathbf{K}\bar{x}^k)$
8: $x^{k+1} = (\mathbf{I} + \tau \partial G)^{-1}(x^k - \sigma \mathbf{K}^* y^{k+1})$
9: $\bar{x}^{k+1} = x^{k+1} + \theta(x^{k+1} - x^k)$
10: **end while**

y, and F^* is the convex conjugate of F obtained from the Legendre-Fenchel transformation.

Algorithm 1 shows the iterative method used to solve the primal-dual problem. Parameters τ,σ and θ are set according to the norm of matrix $\mathbf{K}$. Lines 7 and 8 in the algorithm are calculated by solving the so called resolvent operator given by:

$$y = (\mathbf{I} + \tau \partial F)^{-1}(\tilde{y}) = \arg\min_y \{\frac{\|y - \tilde{y}\|^2}{2\tau} + F(y)\}$$

B. Primal-Dual solution for minimizing $\|\mathbf{Ax} - \mathbf{b}\|_1$

Our objective is to solve the following minimization problem:

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_1$$

comparing this equation with Eq.(2) we can see the following equivalences: $\mathbf{K} \leftrightarrow \mathbf{A}$, $F(\mathbf{Ax}) = \|\mathbf{Ax} - \mathbf{b}\|_1$ and $G(\mathbf{x}) = 0$. Therefore the primal-dual problem is stated as

$$\min_{\mathbf{x}} \max_{\mathbf{y}} \langle \mathbf{Ax} - \mathbf{b}, \mathbf{y} \rangle - F^*(\mathbf{y})$$

where the dual conjugate F^* is the indicator function defined as:

$$F^*(\mathbf{y}) = \begin{cases} 0, & \|\mathbf{y}\|_\infty \leq 1 \\ \infty, & \|\mathbf{y}\|_\infty > 1 \end{cases} \quad (4)$$

For this particular problem, the solution for the primal and dual variables at each k iteration is sketched in algorithm 2. To speed up the convergence we use a diagonal preconditioning for τ and σ explained in [20] in lines 2 to 5. The resolvent operator for the dual variable is given in line 13 whereas for the primal variable the resolvent operator is just the identity (since $G(\mathbf{x}) = 0$). Notice that the algorithm is based on simple sparse matrix-vector products.

IV. $\mathcal{L}_1$-INITIALIZATION FOR POSE GRAPH SLAM

Initialization for non linear optimization is crucial to achieve convergence. A robust back-end solution must provide an accurate initial seed independently of the norm used during global iterations. We use the primal-dual algorithm to generate an accurate seed that will be used later on as input to the $\mathcal{L}_1$ minimizer. In order to achieve the major effectiveness of the primal dual algorithm, we propose to decouple the rotation and translation variables. The advantage of this

Algorithm 2 $\mathbf{x}^*$ = primal_dual_$\mathcal{L}_1$(A,b)

1: { *Calculate preconditioners* }
2: $T = diag(\tau)$ `with` $\tau = (\tau_\mathbf{1}, \ldots \tau_\mathbf{n})$
3: $\Sigma = diag(\sigma)$ `with` $\sigma = (\sigma_\mathbf{1}, \ldots \sigma_\mathbf{m})$
4:
5: $\tau_j = \frac{1}{\sum_{i=1}^m \|A_{ij}\|}$ $\sigma_i = \frac{1}{\sum_{j=1}^n \|A_{ij}\|}$
6:
7: {*Initialize* $x^0_{n\times 1}$, $y^0_{m\times 1}$, $\theta \in [0,\ 1]$ }
8: $\bar{x}^0 = x^0$
9: **while** $k \leq N$ **do**
10: { *Update Dual* }
11: $\mathbf{y}^{k+1} = \mathbf{y}^k + \Sigma \cdot (\mathbf{A}\bar{\mathbf{x}}^k - \mathbf{b})$
12:
13: $y_i^{k+1} = \frac{\tilde{y}_i^{k+1}}{\max(1, \|\tilde{y}_i^{k+1}\|)}$ $\forall y_i \in \mathbf{y}$
14:
15: { *Update Primal* }
16: $\mathbf{x}^{k+1} = \mathbf{x}^k - T \cdot \mathbf{A}^T\mathbf{y}$
17:
18: $\bar{\mathbf{x}}^{k+1} = \mathbf{x}^{k+1} + \theta(\mathbf{x}^{k+1} - \mathbf{x}^k)$
19: $\mathbf{x}^k = \mathbf{x}^{k+1}$
20: **end while**
21: **return** $\mathbf{x}^* = \mathbf{x}_k$

scheme is the convex nature of the resulting cost functions. Algorithm 3 details the initialization process. First, the Jacobian and gradient for the angle linear system $\mathbf{J}_\theta\theta = \mathbf{g}_\theta$ are computed. In this step only the measured relative orientations are involved thus producing a Jacobian pattern whose rows are trivially expressed as $\mathbf{J}_{ij} = [0 \ldots -1 \ldots 0 \ldots 1 \ldots]$. An initial correction of the angles is carried by calling the $primal\_dual\_\mathcal{L}_1$ algorithm as a subroutine. This delivers a set of improved initial orientations θ^*. A similar procedure is applied to solve the translation subproblem $\mathbf{J_t t} = \mathbf{g_t}$. By fixing the orientation to the partial solution θ^*, the decoupled translation problem becomes also convex obtaining an initial solution $\mathbf{t}^*$.

V. $\mathcal{L}_1$ NORM BASED OPTIMIZATION FOR POSE GRAPH SLAM

Global optimization updates using the $\mathcal{L}_1$ cost function are also carried out in a primal-dual sense. Similar to the Gauss Newton method, we provide an iterative method shown in

Algorithm 3 getInitialSeed(FG)

{ *orientation seed* }
$[\mathbf{J}_\theta, \mathbf{g}_\theta] = buildSetupOrientation(FG)$
$\theta^* = primal\_dual\_\mathcal{L}_1(\mathbf{J}_\theta\ ,\ \mathbf{g}_\theta)$
$\mathbf{x} = fixTheta(\mathbf{x}, \theta^*)$

{ *translation seed* }

$[\mathbf{J}_t, \mathbf{g}_t] = buildSetupTranslation(FG)$
$\mathbf{t}^* = primal\_dual\_\mathcal{L}_1(\mathbf{J}_t\ ,\ \mathbf{g}_t)$

algorithm 4. At the current linearization point, we build the Jacobian matrix and gradient vector of our pose factor graph given the state vector $\mathbf{x}^k$ and the set of relatives observations $\mathbf{z}$. The calculation of the gradients and Jacobians is exactly the same as for the $\mathcal{L}_2$ norm based optimization. The state vector is updated using the composition transformation as in the g2o optimizer.

Algorithm 4 Factor Graph Optimization

1: $\{$Given a factor graph $FG(\mathcal{V},\mathcal{E})\}$
2: $\mathbf{x}^0 = getInitialSeed(FG)$
3: **while** $k \leq N_g$ **do**
4: $\quad [\mathbf{J},\mathbf{g}] = buildSetup(\mathbf{x}^k\ ,\ \mathbf{z})$
5: $\quad \mathbf{dx}^k = primal\_dual\_\mathcal{L}_1(\mathbf{J}\ ,\ \mathbf{g})$
6: $\quad \mathbf{x}^{k+1} = \mathbf{x}^k \oplus \mathbf{dx}^k$
7: **end while**

VI. Results

We provide a fast and reliable C++ implementation of our $\mathcal{L}_1$ based Factor graph algorithm running on an Intel Core i7-2630QM CPU at 2.9GHz. Our back-end module allows us to use the $\mathcal{L}_1$ and $\mathcal{L}_2$ based cost functions for primal-dual and NLSQ optimizations respectively. Our implementation is based on the SuitSparse library with efficient sparse matrix-vector operations for $\mathcal{L}_1$, and a fast Cholesky factorization solver with reordering for $\mathcal{L}_2$. We have tested our $\mathcal{L}_2$ NLSQs solver against the g2o software obtaining the same efficiency. Nonetheless, we have run g2o in the expriments to assure the thoughtfulness of the evaluation and to provide fair comparisons with the most popular state of the art optimizer. The algorithms proposed in this paper are run on three well known real datasets widely used: `Intel`, `Manhattan3500` and `City10k`. The simplicity of the $\mathcal{L}_1$ calculations also allows us to carry out more challenging evaluations on larger simulated Manhattan worlds with 10k and 50k nodes.

Our first experiment evaluates the ability of the primal dual algorithm to obtain good initial seeds. Algorithm 3 is executed to find an improved set of angles from which we calculate a set of improved translations. Figures 2a and 2b show the evolution of the cost function per iteration for the big `Manhattan10k` dataset. In both cases, we intentionally over iterate the algorithm using $1e{+}5$ primal-dual iterations until the convex function gets its lowest costs and reaches the optimal minimum. The results empirically show that a maximum of $1e{+}4$ iterations are required to achieve the searched solution. Notice the $O(1/k)$ cost reduction behavior predicted by the theory. Figure 2c shows also the evolution of the cost function during the non-linear global optimization (algorithm 4) with 10000 outer updates and 400 inner primal-dual iterations. Again, we have performed much more iterations than the strictly necessary to get the basin minimum. According to the results, at least 30 outer iterations are required to obtain a solution for the non linear problem. We can avoid the use of a fixed number outer iterations by using similar stop criteria on the gradient norm and cost values as in g2o. Figure 3 shows the final maps

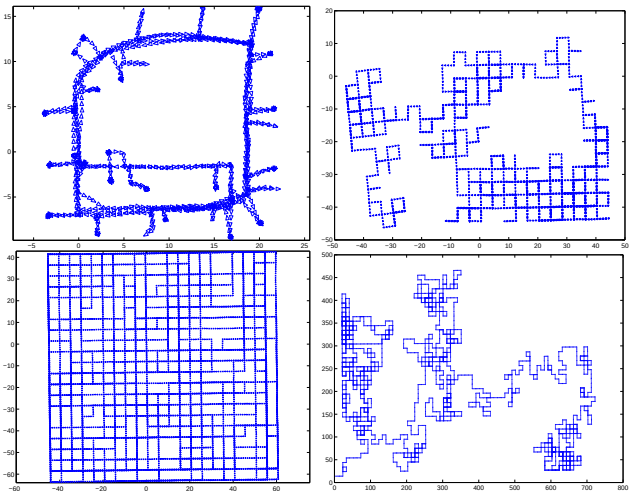


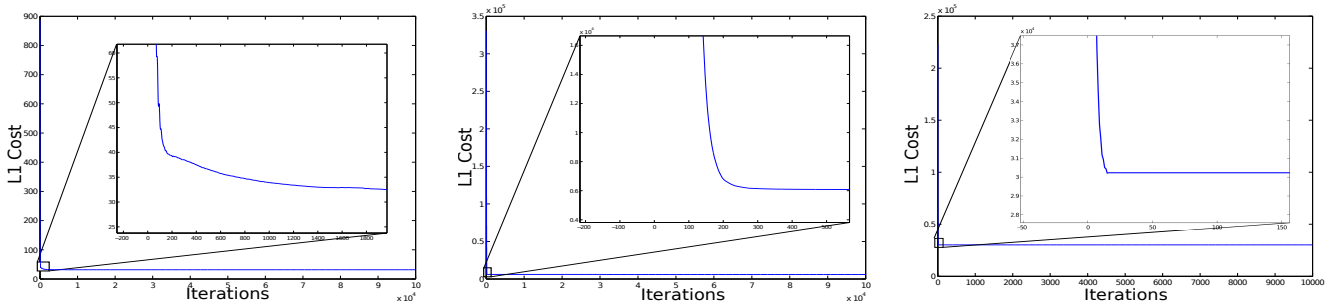
Fig. 3: Optimized maps with $\mathcal{L}_1$ based Factor graph SLAM. Intel (top-left). Manhattan3500 (top-right). City10k (bottom-left). Manhattan10k (bottom-right).

TABLE I: Initial and final $\mathcal{L}_1$ costs

	Initial cost	$\mathcal{L}_1$ solution Final Cost
Manhattan3500	5940	1319,53
Manhattan10k	5.2064e+07	30206,5
Intel	1513,17	982,051
City10k	2.78513e+06	4021,95
Manhatan50k	7.14591e+08	117734

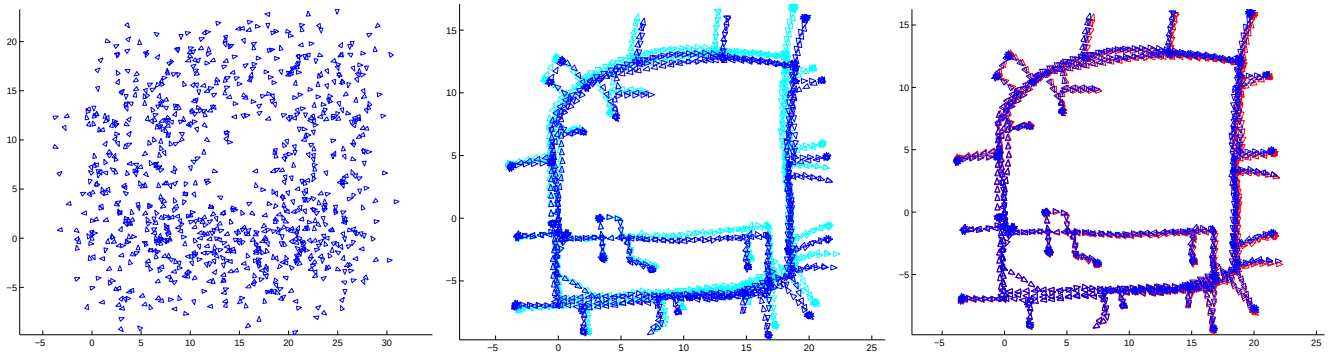
obtained for 4 datasets. In Table I we summarize the initial and final $\mathcal{L}_1$ costs for the evaluated datasets. Figure 4 depicts for the `Intel` dataset the output sequence for each step of the optimization. To clearly show the effect of the initial seed algorithm we generate a very noisy data as input to the optimizer. The improved seed was obtained after 1000 primal-dual iterations for both angles and translations. The final $\mathcal{L}_1$ non-linear optimization required 5 outer iterations with 500 primal dual inner iterations.

Although our algorithm usually requires a larger number of $\mathcal{L}_1$ iterations to achieve a correct map compared to g2o, each iteration requires less time due to the simple sparse matrix-vector multiplications involved. On the contrary, CHOLDMOD based solvers as g2o use much more time per iteration to solve a Newton step. In table II we summarize the time in seconds for each iteration for the different algorithms. For the Manhattan50k dataset, which contains 50000 nodes and 700000 edges, the g2o algorithm gets stacked solving the first iteration while the proposed $\mathcal{L}_1$ algorithm is able to achieve a solution. In fact, the table shows the CHOLMOD dependency on the graph structure. The `City10k` map is solved in just 0.12s while the `Manhattan10k`, which contains the same number of nodes but a different edge distribution takes 2.31s per iteration. An incremental version of the proposed algorithm was tested against the incremental g2o solution. In Figure 5 we show the



(a) (b) (c)

Fig. 2: Results obtained after the execution of our $\mathcal{L}_1$ based primal dual algorithm. The plots show the cost evolution per number of total iterations on the `Manhattan10k` dataset. (a) Cost vs iterations for convex angle correction. (b) Cost vs iterations for convex translation correction. (c) Global cost vs outer iterations for the complete non-linear optimizations. For each plot, a close up picture is taken to show that the minimum cost is achieved during the first iterations.



(a) (b) (c)

Fig. 4: Graph optimization sequence for the `Intel` dataset. (a) Initial seed with added noise. (b) Result after seed initialization (light-cyan) and final $\mathcal{L}_1$ optimization (dark-blue). (c) Comparison of $\mathcal{L}_1$ based solution (blue) vs. $\mathcal{L}_2$ g2o solution (red).

TABLE II: Time per iteration in seconds

	$\mathcal{L}_1$ solution			$\mathcal{L}_2$-g2o solution
	Angle	Traslation	Global	
Manhattan3500	0,0009	0,002	0,003	0,016
Manhattan10k	0,003	0,0075	0,011	2,31
Intel	0,0003	0,0006	0,001	0,0067
City10k	0,003	0,0076	0,013	0,12
Manhattan50k	0,015	0,033	0,056	??

time per iteration required by g2o and our method to solve the City10k dataset. Notice that the behaviour of our algorithm remains unaltered with the number of nodes in contrast to g2o. For more than 5000 nodes the running times of both algorithms is very similar. Finally, we show the robustness of the $\mathcal{L}_1$ based Factor Graph algorithm in presence of strong spurious data. Our algorithm is able to achieve an accurate result even when the datasets contain very incorrect links without any adjustment of new experimental parameters. In Figure 6 bottom, we show an optimized map of the `Intel` and `Manhattan3500` datasets where 10 and 2 links have been randomly added respectively. These links connect far away robot positions whereas the corresponding spurious measurements tells the algorithm that the distance is zero. For this datasets we run g2o using the robust Huber norm with different kernel values. A total of 100 iterations have been done by g2o using a Levenberg Marquardt step. On the right of the figure, we can observe the final distorted solutions. In contrast, $\mathcal{L}_1$ based Factor Graph SLAM uses 10000 iterations for angle and translation initialization with no global optimization obtaining good results. The input noisy data is depicted in green. Random spurious data corresponds to the red links.

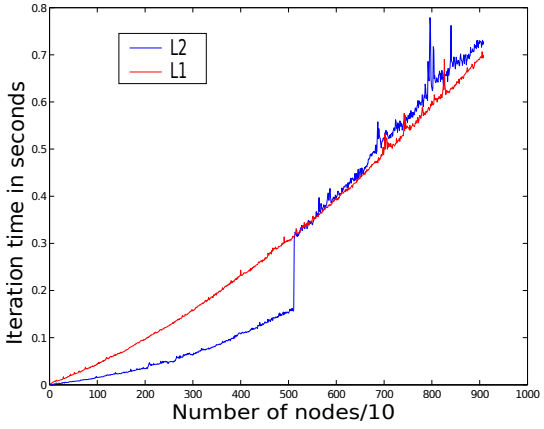


Fig. 5: Time per iteration for an incremental problem wih the `City10k` dataset.

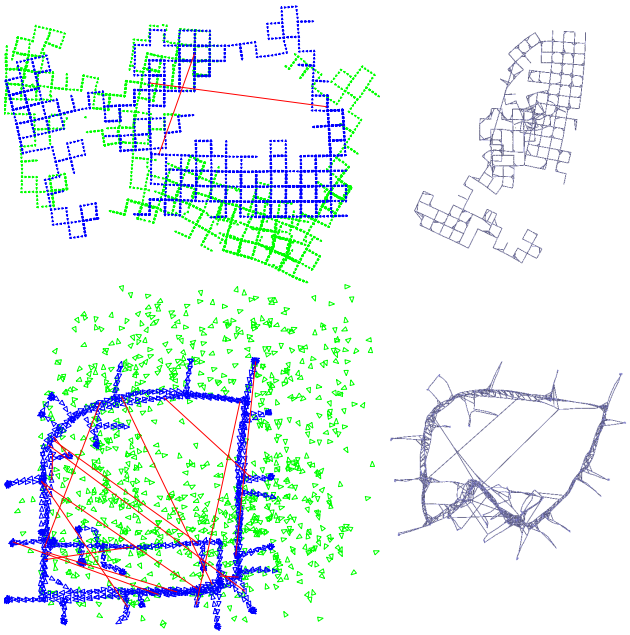


Fig. 6: Brief evaluation of the graph SLAM algorithms under the presence of spurious data. On the left, the solution obtained by the proposed algorithm. On the right, the solution provided by g2o with robust Huber kernel.

VII. Conclusions and Future Work

The current work presents a novel Factor Graph algorithm for pose graph SLAM based on minimizing an $\mathcal{L}_1$ norm cost function instead of the traditional NLSQs scheme. We introduce into the SLAM literature a primal-dual transformation able to cope with non differentiable functions like the $\mathcal{L}_1$ norm in reasonable time. To solve the non-linear and non-differentiable optimization, we propose a double iterative algorithm: On the one hand, an inner iterative subroutine is executed to find the $\mathcal{L}_1$ norm solution of a linear system; On the other hand, an outer iteration is performed, as in traditional Gauss Newton methods, that linearizes the residuals and builds the linear system that will feed the inner loop. As additional contributions, we propose a simple an effcient algorithm to calculate a very good seed for the non-linear optimization based on decoupling the orientation and translation variables to obtain two convex minimization problems easily solved using the same paradigm. Since the primal-dual formulation proposed only performs sparse matrix-vector products, the algorithm is able to reach accurate solutions on very large graph datasets in which state of the art optimizers can not perform well. We have also shown the ability of the proposed optimization method to reject strong spurious data. Unlike the known Huber norm, the $\mathcal{L}_1$ norm is more robust and does not require the tuning of any kernel parameter.

This work opens a bunch of future lines of research. Since all norms are convex, we could make use of the primal dual formulation to introduce new norms into our optimization back-end such as the $\mathcal{L}_\infty$ norm or to combine different norms to derive new algorithms to improve data association during the optimization steps. In a practical sense, we can extent the use of our primal dual algorithm to 3D pose graph optimization. We also feel that other fields of robotic reseach where optimization of non-differentiable functions is required can be drastically favored with this new formulation.

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] J. Nocedal and S. Wright, *Numerical Optimization*. New York: Springer-Verlag, 1999.

[3] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing," *Int. J. Robotics Research*, vol. 25, no. 12, December 2006.

[4] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Fast Incremental Smoothing and Mapping with Efficient Data Association," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Rome, Italy, Apr 2007.

[5] 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*, 2010.

[6] G. Grisetti, D. L. Rizzini, C. Stachniss, E. Olson, and W. Burgard, "Online Constraint Network Optimization for Efficient Maximum Likelihood Map Learning," in *Proc. of the IEEE Inter. Conf. on Robotics and Automation (ICRA)*, 2008.

[7] B. Triggs, P. McLauchlan, R. Hartley, and andrew Fitzgibbon, "Bundle Adjustment – A Modern Synthesis," in *Vision Algorithms: Theory and Practice*, ser. LNCS, W. Triggs, A. Zisserman, and R. Szeliski, Eds. Springer Verlag, 2000, pp. 298–375.

[8] T. A. Davis, *Direct Methods for Sparse Linear Systems*, ser. Fundamentals of Algorithms. SIAM, 2006. [Online]. Available: http://www.cise.ufl.edu/research/sparse/CSparse

[9] Y.-D. Jian, D. C. Balcan, and F. Dellaert, "Generalized subgraph preconditioners for large-scale bundle adjustment," in *Proc. of the 15th Int. Conf. on Theoretical Foundations of Computer Vision: outdoor and large-scale real-world scene analysis*. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 131–150.

[10] E. Olson, J. Leonard, and S. Teller, "Spatially-Adaptive Learning Rates for Online Incremental SLAM," in *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[11] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *In Proc. of Robotics: Science and Systems (RSS*, 2007.

[12] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 3607–3613.

[13] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," in *Proceedings of Robotics: Science and Systems (RSS)*, Sydney, Australia, July 2012.

[14] N. Sünderhauf and P. Protzel, "Switchable constraints for robust pose graph SLAM," in *IROS*, 2012, pp. 1879–1884. [Online]. Available: http://dx.doi.org/10.1109/IROS.2012.6385590

[15] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U. K.: Cambridge University Press, 2000.

[16] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," in *Proc. of the 11th annual Int. Conf. of the Center for Nonlinear Studies on Experimental mathematics : computational issues in nonlinear science*. Elsevier North-Holland, Inc., 1992, pp. 259–268. [Online]. Available: http://dl.acm.org/citation.cfm?id=142269.142312

[17] R. T. Rockafellar, *Convex Analysis*. Princeton, New Jersey: Princeton University Press, 1970.

[18] A. Chambolle and T. Pock, "A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging," *J. Math. Imaging Vis.*, vol. 40, no. 1, pp. 120–145, May 2011. [Online]. Available: http://dx.doi.org/10.1007/s10851-010-0251-1

[19] Y. Nesterov, "Smooth minimization of non-smooth functions," *Math. Program.*, vol. 103, no. 1, pp. 127–152, May 2005. [Online]. Available: http://dx.doi.org/10.1007/s10107-004-0552-5

[20] T. Pock and A. Chambolle, "Diagonal preconditioning for first order primal-dual algorithms in convex optimization," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2011, pp. 1762–1769.