Eliminating Conditionally Independent Sets in Factor Graphs: A Unifying Perspective based on Smart Factors

Luca Carlone, Zsolt Kira, Chris Beall, Vadim Indelman, and Frank Dellaert

***Abstract*— Factor graphs are a general estimation framework that has been widely used in computer vision and robotics. In several classes of problems a natural partition arises among variables involved in the estimation. A subset of the variables are actually of interest for the user: we call those *target* variables. The remaining variables are essential for the formulation of the optimization problem underlying maximum a posteriori (MAP) estimation; however these variables, that we call *support* variables, are not strictly required as output of the estimation problem. In this paper, we propose a systematic way to abstract support variables, defining optimization problems that are only defined over the set of target variables. This abstraction naturally leads to the definition of *smart factors*, which correspond to constraints among target variables. We show that this perspective unifies the treatment of heterogeneous problems, ranging from structureless bundle adjustment to robust estimation in SLAM. Moreover, it enables to exploit the underlying structure of the optimization problem and the treatment of degenerate instances, enhancing both computational efficiency and robustness.**

I. Introduction

Future generations of robots will be required to operate fully autonomously for extended periods of time over large-scale environments. This goal stresses the importance of *scalability* for the estimation algorithms supporting robot navigation. State-of-the-art techniques for localization and mapping (*SLAM*) have reached a maturity that enables fast solution of medium-sized scenarios [1], [2], [3], [4]. These techniques are based on a maximum a posteriori estimation paradigm that computes the optimal state estimate by solving a nonlinear optimization problem. While in specific cases it is possible to exploit problem structure and devise closed-form solutions (or approximations) [5], [6], general techniques are based on iterative nonlinear optimization. The optimal solution of the original nonlinear optimization problem is computed by solving a sequence of linear systems (*normal equations*). In real-world applications the state may involve robot poses, location of external landmarks, and other auxiliary variables (e.g., sensor biases, sensor calibration). Therefore, the optimization problem to be solved is very large. Moreover, the size of the state to be estimated grows over time, and this prevents long-term operation.

This work was partially funded by the National Science Foundation Award 11115678 "RI: Small: Ultra-Sparsifiers for Fast and Scalable Mapping and 3D Reconstruction on Mobile Robots".

L. Carlone, C. Beall, V. Indelman, and F. Dellaert are with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA, {luca.carlone,cbeall3}@gatech.edu, {frank,indelman}@cc.gatech.edu.

Z. Kira is with the Georgia Tech Research Institute, Atlanta, GA 30332, USA, zkira@gatech.edu.

The issue of scalability recently attracted the interest of the research community for its practical relevance. In [4], Polok et al. propose ad-hoc techniques for speeding-up solution of normal equations, exploiting problem structure. In [7], Ila et al. consider the problem of *pose graph optimization* and propose information-theoretic measures to select only the most informative edges, so to prevent uncontrolled growth of the graph. In [8], Stachniss and Kretzschmar propose a *graph compression* technique for the specific case of laser-based SLAM. In [9], Huang et al. focus on consistency issues and propose a node marginalization scheme, based on Schur complement; moreover, they introduce a technique for edge sparsification, based on ℓ_1-regularized optimization. In [10], Carlevaris-Bianco and Eustice propose a sparsification technique based on marginalization for pose graphs.

Scalability issues also emerge when dealing with *outlier* rejection in SLAM. Outlier rejection can be addressed in the *front-end* (i.e., before performing inference) [11], [12] or the choice of removing an outlying measurement can be part of the inference process [13]. The latter approaches are becoming very popular as they allow the optimization process to dynamically decide the best set of measurements; these approaches have been shown to be resilient to extreme percentages of outlying measurements. Sünderhauf and Protzel [13] propose to model the choice of selecting or discarding a measurement using binary variables that are added to the optimization problem; relaxation is then used to solve the otherwise intractable mixed-integer optimization problem. In this case, the price of robustness is the increase in the computational cost of optimizing over a large number of *latent* variables (the binary variables). Olson and Agarwal [14] proposes a max-mixture model, to avoid introduction of binary variables. Agarwal et al. [15] propose to fix the value of the relaxed binary variables, in order to circumvent the solution of a large optimization problem.

While recent literature offers effective solutions for specific problem instances (e.g., pose-only graphs, sensor-specific applications, outlier rejection), in this context we are interested in developing a *general probabilistic* framework for variable elimination. The basic observation is that in several computer vision and robotics problems a natural partition arises in the set of variables. In particular, it is possible to split the set of variables into two sets: a first set, that we call *target* variables, contains variables that are actually required as output of the estimation problem. The remaining variables, which we call *support* variables, are only functional to the estimation problem although they are often not required as output of the estimator. Furthermore,

the set of support variables can often be partitioned into subsets that are conditionally independent given the set of target variables. Several motivating examples in which this pattern emerges are reported in Section III. Starting from this observation we model the problem using *factor graph* formalism. We then show how to reduce the original optimization problem to a smaller problem, which only involves the (usually small) set of target variables, abstracting support variables (*implicit* parameterization). Moreover, we show that elimination of the conditionally independent subsets of support variables naturally leads to the definition of new factors, that we call *smart factors*, which substitute a (usually large) set of factors that were previously connected to the support variables. The new factors are "smart" for different reasons: first of all, thanks to the implicit parametrization, the optimization problem to be solved becomes smaller (small number of variables and small number of factors) hence the introduction of these factors leads to a computational advantage; moreover, these factors can easily include *domain-specific knowledge,* allowing the management of degenerate problem instances (e.g., under-constrained problems) without compromising the solution of the overall problem. Finally, a smart factor contains the intelligence to retrieve a subset of the support variables, if needed, exploiting the conditional independence property that is satisfied by these variables.

The contribution of the paper is threefold: (i) a *tutorial* contribution, as we propose a unifying perspective that draws connections among techniques applied to different problems and in different research communities (e.g., robotics and computer vision); (ii) a *practical* contribution, as we show that the definition of smart factors leads to improvements in the estimation process in terms of computational effort and robustness; (iii) a *technical* contribution, as we discuss several elimination techniques and we show insights on their formulation and on the underlying approximations; for instance we show the equivalence of the Schur complement (widely used in computer vision) and the *null space trick* which has been proposed in the context of EKF-based navigation. Moreover, we show that in *rotation graphs* (graphs in which only a rotation matrix is assigned to each node) elimination can be performed nonlinearly, while the same approach constitutes an approximation for *pose graphs*. The source code of several smart factors is available online at `https://borg.cc.gatech.edu/`.

The paper is organized as follows. Section II provides preliminaries on factor graphs. Section III describes some motivating examples. Section IV introduces the probabilistic framework, the concept of *implicit parametrization*, and shows how the elimination of support variables leads to the definition of smart factors. Section V provides viable techniques for elimination, for each of the motivating examples. Conclusions are drawn in Section VI.

II. Factor Graphs

Factor graphs are a general representation of estimation problems in terms of graphical models [16]. A factor graph is a bipartite graph $\mathcal{G} = \{\mathcal{F}, \Theta, \mathcal{E}\}$, where $\mathcal{F}$ is the set of *factor nodes*, Θ is the set of *variable nodes*, and $\mathcal{E}$ are edges in the graph. A (given) measurement Z_i is associated to the factor node $\phi_i \in \mathcal{F}$. An (unknown) variable is associated to each variable node in Θ. Fig. 1(a) provides a pictorial representation of a factor graph. The graph is bipartite, in the sense that edges may only connect a node in $\mathcal{F}$ to a node in Θ. A factor graph essentially encodes a density over the variables in Θ:

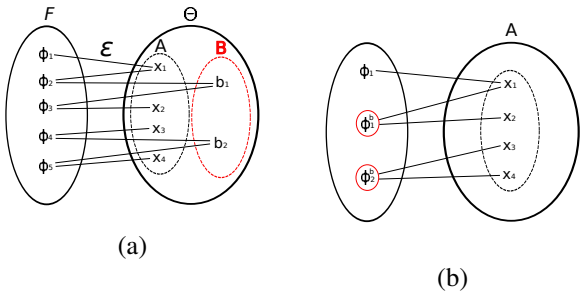
$$P(\Theta|Z) = \prod_{i=1}^{|\mathcal{F}|} P(\Theta_i|Z_i) = \prod_{i=1}^{|\mathcal{F}|} \phi_i(\Theta_i) \tag{1}$$

where each *factor* $\phi_i(\Theta_i) = P(\Theta_i|Z_i)$ involves a subset of variables $\Theta_i \subseteq \Theta$. Note that Bayes nets and Markov random fields can be modeled as factor graphs; moreover, factor graphs also find applications outside the estimation domain (e.g., constraint satisfaction problems). In this context we focus on those factor graphs where $\phi_i(\Theta_i)$ can be normalized to a probability distribution, i.e., $\int \phi_i(\Theta_i) \mathrm{d}\Theta_i = 1$; clearly this can easily accommodate discrete variables and the mixed continuous-discrete case.

Given a factor graph $\mathcal{G} = \{\mathcal{F}, \Theta, \mathcal{E}\}$, the *maximum a posteriori estimate* of variables in Θ is defined as:

$$\Theta^\star = \arg\max_{\Theta} \prod_{i=1}^{|\mathcal{F}|} \phi_i(\Theta_i) = \arg\min_{\Theta} \prod_{i=1}^{|\mathcal{F}|} -\log \phi_i(\Theta_i) \tag{2}$$

where $\phi_i(\Theta_i)$ is usually approximated as a Gaussian density and $-\log \phi_i(\Theta_i)$ becomes a squared residual error.



(a) (b)

Fig. 1. (a) Factor graph with 5 factors $\mathcal{F} = \{\phi_1, \ldots, \phi_5\}$ and 6 variable nodes $\Theta = \{x_1, \ldots, x_4, b_1, b_2\}$. In this paper we discuss how to eliminate a subset of the original variables. The elimination leads to the creation of a new graph (b), which contains a smaller number of variables and a small number of *smart* factors (factors with the red circle in (b)).

III. Motivating Examples

In several computer vision and robotics problems a natural partition arises in the set of variable nodes Θ. In particular, it is possible to write $\Theta = \{A, B\}$, where A is the set of *target* nodes, that are actually required as output of the estimation problem. The remaining nodes B are only functional to the estimation problem although they are often not required as output of the estimator; we call B the *support* nodes. Furthermore, in many practical applications the set B naturally splits in (small) conditionally independent sets. In the following we discuss many practical examples in robotics and computer vision in which one can observe this property.

Long-term navigation. Consider the pose graph in Fig. 2(a). The robot starts at pose P_o and acquires odometric measurements which are modeled as factors connecting consecutive poses. Moreover, the robot can occasionally acquire loop closing constraints (dashed blue lines in the figure), using exteroceptive sensors (e.g., camera, GPS). Odometric measurements are acquired at high rate (e.g., 100Hz, from an *Inertial Measurement Unit*). However, one is usually interested in having estimates at a lower frequency (say, 1Hz). For instance, in the example in Fig. 2(a) we may only want to estimate few poses, say P_0, P_1, P_2, P_3, P_4. In our framework we treat $A = \{P_0, P_1, P_2, P_3, P_4\}$ as target variables and abstract the remaining poses as support variables into *smart factors*. The resulting graph is the one in Fig. 2(b), where the smart factors are shown as red dotted links. We note that the choice of the target variables implies the following conditional independence relation:

$$P(\Theta) = P(A) \prod_{b_i \subset B} P(b_i|A) \tag{3}$$

where the set B (support variables) includes all poses that are not in A, and each subset b_i contains the nodes along the path connecting two target variables; the subsets b_i are conditionally independent given the separator nodes A.

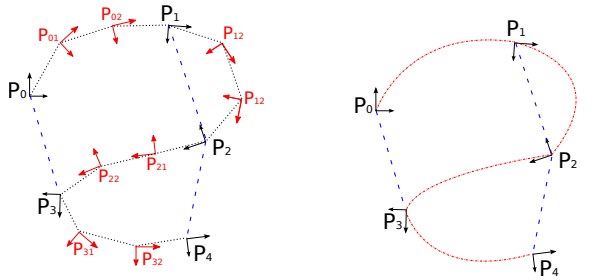


Fig. 2. Long-term navigation example. (a) Sensor measurements are acquired at high rate and estimation over the entire graph rapidly becomes prohibitive. In this paper we define a set of target variables $A = \{P_0, P_1, P_2, P_3, P_4\}$, and eliminate the remaining poses (support variables) leading to the smaller graph in (b).

Landmark-based SLAM. Consider the range-based SLAM problem in Fig. 3. A mobile robot takes range measurements with respect to external landmarks and has to solve SLAM using these measurements and measurements of the ego-motion from odometry. In order to perform estimation of robot motion one usually solves a large optimization problem involving both robot poses (target variables) and the position of external landmarks (support variables). This problem can be not well-behaved, as, for instance, the position of a landmark is ambiguous for $m < 3$ range measurements. If we call R the set of robot poses and $L = \{l_1, \ldots, l_m\}$ the set of landmarks positions, we have the following conditional independence relation

$$P(R, L) = P(R) \prod_{l_i \in L} P(l_i|R)$$

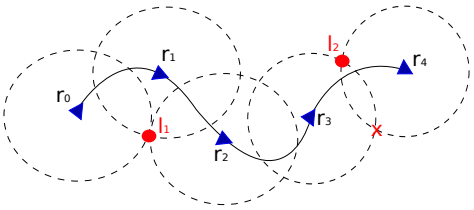


Fig. 3. Landmark-based SLAM. A mobile robot takes range measurements with respect to external landmarks and has to solve SLAM using these measurements and measurements of the ego-motion from odometry. Note that in some cases the landmark position remains ambiguous, e.g., from two range measurements taken from pose r_3 and r_4 the robot cannot determine if the landmark l_2 is in the position denoted with a red dot, or the one denoted with the red cross.

in which each landmark is independent on all the other landmarks given robot poses.

Exactly the same pattern emerges for the case in which the robot observes landmarks from a vision sensor *(bundle adjustment* in computer vision). Landmarks are only used to properly model each measurement in terms of projective geometry, although they may not be useful for the final user (instead of a sparse map of features the user may be interested in dense map representations which can be easily built from robot poses). In our framework we eliminate landmarks, generating smart factors connecting robot poses.

Structure from motion. In this case the user is interested in the structure, hence in landmark positions rather than observation poses. A typical example of this problem is *photo tourism* [17], in which one wants to reconstruct the 3D geometry of a place from a collection of photos. In this case the camera calibration has to be estimated, therefore we have three sets of variables: the camera poses C, the calibration matrices K, and the landmark positions L. In SFM we can observe conditional independence relations between different sets. For example we can factorize the joint density as:

$$P(C, K, L) = P(C, K) \prod_{l_i \in L} P(l_i|C, K) =$$
$$= P(C)P(K|C) \prod_{l_i \in L} P(l_i|C, K)$$

Note that the other way, eliminating K first, create correlation between all landmarks and cameras, destroying conditional independence.

Outlier rejection in SLAM. Let us consider a setup in which a mobile robot has to perform SLAM in presence of outliers [13], [14], [15]. In this case, some of the measurements are distributed according to a given measurement model (say, a zero-mean Gaussian noise), while other measurements are *outliers*. Recent approaches propose to introduce a binary variable for each measurement, with the convention that if the binary variable is equal to 1 the measurement is an inlier, or it is zero for outliers. Also in this case the binaries are conditionally independent given graph configuration. Moreover, the user may be not interested in the binaries, but the estimation problem needs to be solved over these variables as well, as a robust solution requires distinguishing inliers from outliers. In our framework the binary variables do not appear in the optimization and they are implicitly modeled inside smart factors.

IV. Eliminating Conditionally Independent Sets

Suppose we can divide Θ into A and B such that

$$P(\Theta) = P(A) \prod_{b_i \subset B} P(b_i|A) \tag{4}$$

i.e., the support set B can be partitioned into subsets b_i that are *conditionally independent given* the set A (we omit the dependence on the measurements for simplicity); we observed in Section III that this pattern occurs in several practical examples. Note that we can calculate

$$P(b|A) = \prod_{i \in I_i^b} \phi_i(b|A_i) \tag{5}$$

where I_i^b are the indices of factors connected to variables in the i-th support set b_i, and A_i are the separators. Likewise,

$$P(A) = \prod_{i \in I_A} \phi_i(A_i) \times \prod_{b_i \subset B} \phi_i^b(A_i^b) \tag{6}$$

where I_A are the indices of the factors that do not involve any variable in B and each factor $\phi_i^b(A_i^b)$ on the separators is formed by the "sum-product":

$$\phi_i^b(A_i^b) = \int_{b_i} \prod_{k \in I_i^b} \phi_k(A_k, b_i) \mathrm{d}b_i \tag{7}$$

Note that the factors $\phi_i^b(A_i^b)$ only involve a subset of target variables $A_i^b \subseteq A$, which are those variables that shared some factor with the support variables b_i in the original graph. With this we get the marginal density on A, from which we can compute the MAP estimate of the variables in A as:

$$A^\star = \arg\max_A \prod_{i \in I_A} \phi_i(A_i) \times \prod_{b_i \subset B} \phi_i^b(A_i^b) \tag{8}$$

Computing $A^\star$ involves solving an optimization problem over some of the original factors ($\phi_i(A_i)$, with $i \in I_A$) and other new factors $\phi_i^b(A_i^b)$, that we call *smart factors*. Problem (8) is now defined over the set of variables of interest, A, while the original problem (2) was defined over $\Theta = \{A, B\}$; this is important since typically $|B| \gg |A|$. Moreover, the number of smart factors is equal to the number of subsets $b_i \subset B$, and this is usually much smaller number than the original number of factors (see Fig. 1). Later in this section we stress that this perspective has advantages in terms of graph *compression* (reducing the number of variables) and also provides tools to enhance robustness.

Note that in several applications we are not interested in computing the marginal probability (6), but we rather want to compute its maximum $A^\star$. This suggests a second optimization-based elimination approach, which is often a better option with respect to the integration in (7):

$$\begin{aligned} A^\star = \arg\max_{A,B} \textstyle\prod_{i \in I_A} \phi_i(A_i) \prod_{i \in I_B} \phi_i(A_i, b_i) = \\ \arg\max_A \textstyle\prod_{i \in I_A} \phi_i(A_i) \left(\max_B \prod_{i \in I_B} \phi_i(A_i, b_i)\right) = \\ \textstyle\prod_{i \in I_A} \phi_i(A_i) \times \prod_{b_i \subset B} \phi_i^b(A_i^b) \end{aligned} \tag{9}$$

with $\phi_i^b(A_i^b) = \max_{b_i} \prod_{k \in I_i^b} \phi_k(A_k, b_i)$. In (9) we first split the factors including support variables (factors in the set I_B) from remaining factors (set I_A); then we observed that we can solve independently each sub-problem $\max_{b_i} \prod_{k \in I_i^b} \phi_k(A_k, b_i)$ involving a single set of support variables b_i; each optimization sub-problem returns a function of A, i.e., $\phi_i^b(A_i^b) = \max_{b_i} \prod_{k \in I_i^b} \phi_k(A_k, b_i)$. With slight abuse of notation we use the same symbol $\phi_i^b(A_i^b)$ for both elimination techniques (sum-product and optimization-based). Both elimination techniques allows transforming a collection of factors into a small set of smart factors $\phi_i^b(A_i^b)$ whose number is equal to the number of conditionally independent subsets in B. In Section V we tailor these elimination techniques to practical examples. In particular, we discuss cases in which one can perform elimination linearly, nonlinearly, or via upper-bound approximations. Note that after computing $A^\star$ we can easily compute (if needed) the MAP estimate for variables in b_i:

$$b_i^\star = \arg\max_{b_i} \prod_{k \in I_i^b} \phi_k(A_k^\star, b_i) \tag{10}$$

Smart factors. We used the name *smart factors* for the factors $\phi_i^b(A_i^b)$ arising from the elimination of each subset $b_i \in B$. The new factors are "smart" for different reasons: first of all, thanks to the implicit parametrization, the optimization problem to be solved becomes smaller (small number of variables and small number of factors) hence leading to a computational advantage. Moreover, the elimination process to be carried out inside the factor, as per (8) or (9), always has the same structure, therefore the corresponding code can be highly optimized and it is suitable for *cache-friendly* implementations. Furthermore, since the integral (or the maximization) to be solved inside each smart factor is independent on the other factors in the graph, the code can therefore be highly parallelized.

A second reason is that these factors can easily include *domain-specific knowledge,* allowing management of degenerate problem instances (e.g., under-constrained problems) without compromising the overall solution. For instance, in vision-based applications, the marginalization (8) has to follow a different route for *degenerate* robot motion (e.g., pure rotation). This awareness, which is hard to maintain at the factor graph level, becomes trivial if included in a factor that decides the marginalization approach dynamically.

Finally, smart factors contain the intelligence to retrieve a subset of the support variables, if needed, exploiting the conditional independence property that is satisfied by these variables, once the target variables have been computed. In specific applications, the maximization (10) can be computed or approximated in closed form, hence enabling quick retrieval of the complete set of variables $\Theta^\star = \{A^\star, B^\star\}$.

V. Linear and Nonlinear Elimination: Theory and Examples

In practice, the integral (or the maximization) underlying the definition of a smart factor does not have a closed-form solution, but there are lucky cases in which this is possible. In general, it is possible to approximate the expression of the factor locally, or using suitable upper bounds. We will

discuss several examples in which we can exploit closed-form solutions, approximations, or resort to upper bounds.

A. Elimination in Linear(ized) factor graphs

Let us consider the case in which we have linear Gaussian factors connected to the set of support variables b_i. Then we want to compute the expression of the smart factor $\phi_i^b(A_i^b) = \max_{b_i} \prod_{k \in I_i^b} \phi_k(A_k, b_i)$. According to standard procedures, we are rather interested in the (negative) logarithm of the probability density $\phi_i^b(A_i^b)$ (see also eq. (2)):

$$-log\phi_i^b(A_i^b) = \min_{b_i} \sum_{k \in I_i^b} -log(\phi_k(A_k, b_i)) \qquad (11)$$

For the case in which the factors $\phi_k(A_k, b_i)$ are linear Gaussian factors, we can rewrite (11) explicitly as:

$$\min_{x_b} \sum_{k \in I_i^b} \left\| H_a^k x_a + H_b^k x_b - r_k \right\|^2 = \min_{x_b} \| H_a x_a + H_b x_b - r \|^2 \quad (12)$$

where we stack all the subsets of variables in A_k, with $k \in I_i^b$, into a single column vector x_a (for instance in landmark-based SLAM this vector may include all robot poses from which the same landmark b_i is observed); similarly, we stack in a vector x_b the support variables in the set b_i (in our landmark-based SLAM example x_b may describe the position of a single landmark b_i); finally, H_a, H_b are matrices describing a linear(ized) version of the measurement model and r is the vector of residual errors.

From (12) we can easily compute the minimum with respect to x_b, as $x_b^\star = \left(H_b^T H_b\right)^{-1} H_b^T (r - H_a x_a)$. Plugging $x_b^\star$ back into (12) we get the expression of our smart factor:

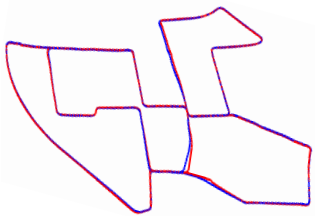
$$-log\phi_i^b(A_i^b) = \left\| \left(I - H_b \left(H_b^T H_b\right)^{-1} H_b^T \right) (H_a x_a - r) \right\|^2 \quad (13)$$

In our framework this sub-problem is solved by the i-th smart factor. We do not claim the novelty of this derivation: the expert reader will notice that the elimination procedure applied here corresponds to the Schur complement, which allows the solving of linear system in a subset of variables. Similar results can be obtained by applying Householder transformations when eliminating b_i in a linear factor graph. In the linear case, the elimination (7) leads to the same result. Less trivial is the fact that one can directly obtain the smart factor by multiplying the terms inside the squared cost (12) by a matrix U_b, which is unitary (i.e., $U_b^T U_b = I$), and defines a basis for the left nullspace of H_b:

$$-log\phi_i^b(A_i^b) = \left\| U_b^T \left(H_a x_a + H_b x_b - r \right) \right\|^2 = \left\| U_b^T \left(H_a x_a - r \right) \right\|^2 \quad (14)$$

While we omit the proof here, it can be shown that (13) and (14) are equivalent. Techniques based on (14) have been used in EKF-based navigation, e.g., [18], [19].

When the constraints (12) are linearized nonlinear constraints, the smart factors (13) (or the equivalent version (14)) have to be recomputed at each iteration of the optimization algorithm. In our smart factor perspective we only use linearization points for the variables in A, while we compute the corresponding linearization points for variables in B via (10), where we use the current linearization point instead of $A_k^\star$. This approach, which may seem to require extra computation, has been shown to improve convergence speed [20] and avoids committing to a specific parametrization of the support variables (our smart factors can dynamically change the structure of the optimization problem (10), enabling correct management of degenerate problem instances).



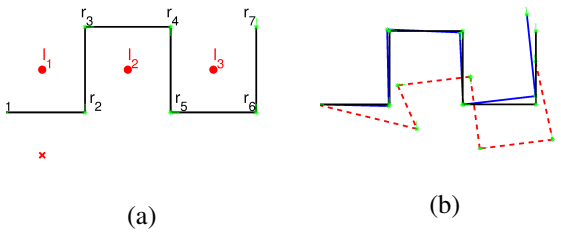
	Original	Smart factors
#Variables	388K	4541
#Factors	1.65M	389K
Time	673.31s	79.13s
Error	0.24%	0.27%

Fig. 4. Application of smart projection factors on the Kitti dataset. The original dataset has 4541 poses and 389008 landmarks. The figure shows the estimated trajectory with standard projection factors (solid red line) versus the one estimated with smart projection factors (solid blue line), with translation error reported as a percentage of distance traveled.

Example 1 - Smart Vision Factors: We apply the linearized elimination approach to a standard vision-based SLAM problem. The data is from the Kitti dataset [21]: a vehicle trajectory is estimated using monocular vision. A standard approach would proceed as follows: one creates a factor graph including both vehicle poses and landmark positions. Then, each monocular observation is modeled as a *projection factor*. The corresponding statistics are shows in Fig. 4 (Column: "Original"). In our approach, each smart factor independently performs elimination of a landmark via (14); then, we only optimize a smaller graph which involves vehicle poses and *smart factors*, see Fig. 4 (Column: "Smart factors"). The same statistics reveal a consistent computational advantage in using the proposed approach.

We remark that Schur complement is a common tool in computer vision [22], [20]. However, smart factors do more than Schur complement: they are able to manage degenerate problems, exploiting domain-specific knowledge. In monocular SLAM one can have the following degeneracies: (i) single observation of a landmark, (ii) degenerate motion (e.g., pure rotation, motion in the direction of a landmark). In both cases one is not able to determine the distance from a landmark; this causes numerical issues in factor graphs with standard projection factors since a variable (3D position of the landmark) is underconstrained. In our formulation the smart factor can detect degenerate instances when solving the subproblem (10) to compute the linearization point for the landmarks. After the degeneracy is detected the smart factor only has to use a different expression of Jacobians in eq. (13), while the overall elimination scheme remains the same. It is easy to show that in degenerate instances, the Jacobians will lead to rotation-only constraints among poses.

Example 2 - Smart Range Factors: We now resume the range-SLAM example of Section III. We consider a toy example to remark that smart factors allow a better management of degenerate problem instances. The example is the one in Fig. 5(a): the robot starts from pose r_1 and traverses the scenario, reaching pose r_7. At each pose, the robot takes range measurements of the position of unknown landmarks in the scenario (l_1, l_2, l_3). Moreover, it takes noisy measurement of the ego-motion (odometry).



(a) (b)

Fig. 5. (a) Toy example of range SLAM. Robot poses: r_1 to r_7; landmark positions l_1 to l_3. (b) Ground truth (solid black line) versus (i) trajectory estimated using standard range factors (dashed red line), and (ii) trajectory estimated using smart range factors (solid blue line).

In our toy example, after the first two time steps (r_1 and r_2) the robot only has two range measurements of l_1, therefore the position of the landmark is ambiguous (in Fig. 5(a) we draw as a red cross the other admissible position, while the actual one is the red dot). In factor graphs with standard range factors, one has to be very careful to the linearization point of the landmark. If at the second time step the landmark is initialized incorrectly (to the red cross, instead of the red dot) the trajectory estimate converges to a local minimum (dashed red line in Fig. 5(b)) and it is not able to recover even if the remaining landmarks (l_2 and l_3) are initialized at the correct position. Conversely, when using smart factors, the factor can detect degeneracy and impose different constraints among poses. The estimate using a basic version of the smart range factors is shown as a solid blue line in Fig. 5(b).

B. Nonlinear Elimination

If the linear factor graphs of the previous section are obtained by linearization, the elimination inside the smart factor has to be repeated whenever the linearization point for the variables in A changes. In few lucky examples the elimination can be done nonlinearly, which implies that the smart factor has to be computed only once.

Example 3 - Smart Rotation Factors: We now show that the elimination framework that we applied in the previous section in a linear factor graph, can be applied *nonlinearly*, in some specific problem instances. We consider a *rotation graph*, which is a factor graph in which each variable is a rotation matrix (in SO(2) or SO(3)), and the factor nodes correspond to relative rotation measurements among nodes pairs. Fig. 2 is an example of this graph for the case in which we disregard the position information and we only consider the orientation of each node. In the example of Fig. 2 we defined a set of target variables, while the support variables corresponds to nodes along the path connecting a pair of target nodes. In this section we want to show that the nodes in each branch can be eliminated into a single smart factor; for this purpose, consider the branch between an arbitrary pair of target nodes u and v (Fig. 6).

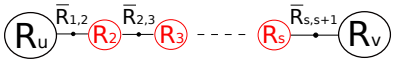


Fig. 6. A branch of a rotation graph; target variables are in black, support variables in red, and relative rotation measurements are denoted with bar.

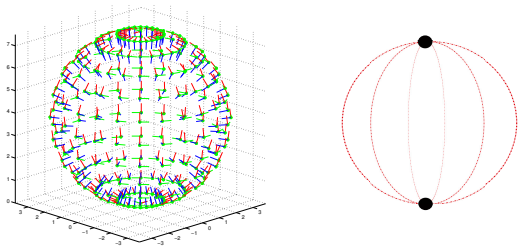
The original factor graph contains s factors along the branch from u to v. The rotations R_u and R_v are target variables. Rotations $R_2, \dots, R_s$ are support variables and will be eliminated. As in Section V-A we start from the negative logarithm of the original factors, which in the rotation case assumes a more complex (nonlinear) form:

$$-\log \phi_i^b(A_i^b) = \min_{R_{1,2},\dots,R_{s,s+1}} \sum_{i=1}^{s} \left\| \mathrm{Log}\left(\bar{R}_{i,i+1}^T R_i^T R_{i+1}\right) \right\|^2_{\sigma^2_{i,i+1}} \quad \text{with} \quad R_1 = R_u \text{ and } R_{s+1} = R_v \tag{15}$$

where R_i is the (unknown) rotation of node $i = 1, \dots, s+1$, $\bar{R}_{i,i+1}$ is the *measured* rotation between R_i and R_{i+1} (which is assumed to be affected by isotropic noise with variance $\sigma^2_{i,i+1}$), and $\mathrm{Log}(R)$ denotes the logarithmic map which returns an element of the Lie algebra whose exponential is R; with slight abuse of notation $\mathrm{Log}(\cdot)$ returns a rotation vector instead of a skew symmetric matrix. In Appendix we prove that nonlinear elimination of R_i, with $i = 2, \dots, s$ leads to a single factor on the target variables R_u and R_v:

$$-\log \phi_i^b(A_i^b) = \left\| \mathrm{Log}\left(\bar{R}_{1,s+1}^T R_u^T R_v\right) \right\|^2_{\sum_{i=1}^{s} \sigma^2_{i,i+1}} \tag{16}$$

where $\bar{R}_{1,s+1} = \bar{R}_{1,2} \dots \bar{R}_{s,s+1}$ is the composition of the rotation measurements in the factors along the branch. The result may appear trivial: the smart factor is nothing else than the composition of the intermediate rotations and the noise variance is the sum of the variances of the intermediate measurements. However, recall that we are in a nonlinear domain, and intuitions that are true for the linear case often fail: for instance, the same elimination procedure cannot be repeated in the case with non-isotropic noise. Also, in a *pose* graph, a similar elimination procedure only results in an approximation, as we briefly show in the next section. We conclude this section by noting that the rotation example is not just a toy problem: from [5] we know that we can use rotations to bootstrap pose estimation and enhance convergence speed and robustness. In Fig. 7 we report an example of use of the smart rotation factors in a spherical scenario, where the origin at the bottom is fixed. The original graph has 266 nodes and factors are connecting nearby nodes along each meridian (Fig. 7(a)); we can convert each branch connecting the node at the bottom with the node at the top into a smart factor, obtaining a graph with only 2 nodes Fig. 7(b). In the table (Column: "SO(3)") we report the mismatch between the rotation estimate at the top node for the complete graph of Fig. 7(a), versus the corresponding estimate in Fig.



Rotation Noise (deg)	SE(3)	SO(3)
0	0	0
5	0.08	< 1E-7
10	0.12	< 1E-7
15	0.23	< 1E-7

Fig. 7. Application of smart rotation factors. (a) Original graph. (b) Compressed graph with smart rotation factors. The table reports the rotation mismatch (norm of the rotation error) of the top-most node in the original versus the corresponding estimate in the compressed graph, for different values of rotation noise. While the estimates coincide for rotation graphs (column: "SO(3)"), they are different for pose graphs (column: "SE(3)").

7(b). The mismatch is negligible and independent on the rotation noise, confirming the derivation in this section. If we repeat the same experiment including the full poses for each node and we consider relative pose measurements, the estimates will no longer match, as discussed in the following.

Example 4 - Smart Pose Factors: Without going in the mathematical details, it is pretty easy to see that, in a *pose graph*, simple pose composition is not equivalent to nonlinear elimination, see also [10]. For this purpose consider Fig. 6 and imagine that a pose is attached to each node, and that we want to simply compose the intermediate poses to obtain a relative pose measurement between node u and v. Applying pose composition and a first-order propagation of the corresponding covariance matrix one may easily realize that both the measurement between u and v (and the corresponding covariance) will depend on the linearization point of all the intermediate nodes. This implies that, as for the linearized case, at every change of the linearization point, the smart factor has to be recomputed, hence the elimination can only be performed on a linearized version of the factors.

Nevertheless one may be satisfied with an approximation and not recompute the smart pose factor (for small changes of the linearization point). This idea appeared several times in literature [23], [24], [25], (while to the best of our knowledge the corresponding elimination in rotation graphs has not been investigated). In particular, in [25] intermediate pose relations (computed from an IMU sensor) are "summarized" or preintegrated into a single constraint. Our C++ implementation of the preintegrated IMU factors, and a Matlab example on the Kitti dataset [21] are available on the website `https://borg.cc.gatech.edu/`.

C. Upper bounds Approximations

In several applications, it is difficult to solve the optimization problem (9) or to compute the integral (8). Therefore, one rather looks for a lower bound for (8), and iteratively optimizes this lower bound. This idea is an old one [26]. In this case our smart factor has to compute

$$\check{\phi}_i^b(A_i^b) \leq \phi_i^b(A_i^b) = \int_{b_i} \prod_{k \in I_i^b} \phi_k(A_k, b_i) \mathrm{d}b_i \qquad (17)$$

It is shown in [26] (and the reference therein) that taking the expectation of the logarithm of $\prod_{k \in I_i^b} \phi_k(A_k, b_i) db_i$ with respect to b_i yields a lower bound for $\phi_i^b(A_i^b)$. Therefore the following *upper* bound on the negative logarithm follows:

$$-\log \phi_i^b(A_i^b) \leq -\log \check{\phi}_i^b(A_i^b) = -\mathop{\mathbb{E}}_{b_i} \left[\sum_{k \in I_i^b} \log \phi_k(A_k, b_i) \mathrm{d}b_i \right] \qquad (18)$$

This is particularly convenient for the case in which the support variables b_i are discrete, as the expectation can be computed as a weighted sum. This approach leads to standard *expectation-maximization (EM)* algorithms, see [26].

Example 5 - Robust SLAM: Many recent robust SLAM approaches follows the philosophy described in Section III (paragraph: *Outlier rejection in SLAM*). An excellent example is [13]: binary variables are used to decide whether to accept or discard a measurement; in [13] binary variables are relaxed to continuous variables in the interval $[0, 1]$. Although the approach [13] greatly enhance robustness to outliers it pays the price of optimizing over a large number of *latent* variables. If the binaries are relaxed to continuous variable the elimination approach that we described in Section V-A is directly amenable to transform the original problem into one that only includes the target variables. However, in this example, we avoid relaxation and maintain discrete values for the support variables. Therefore, we use the elimination technique discussed in Section V-C.

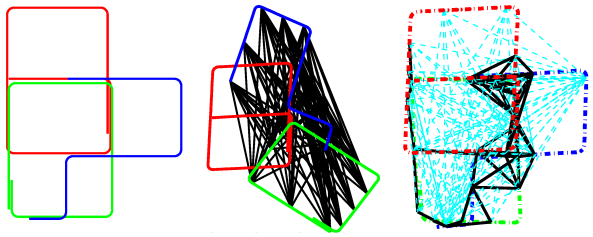


Fig. 8. (a) Ground truth trajectories of 3 robots moving in the same scenario. (b) Tentative correspondences between robot observations (black lines). (c) Trajectories estimated with the elimination approach of Section V-C; after optimization we can distinguish inliers (black solid lines) from outliers (cyan dashed lines).

We consider a multi robot problem in which 3 robots traverse the same scenario (Fig. 8(a)), without prior knowledge of their initial pose and without being able to directly measure relative poses during rendezvous. During operation, the robots try to establish possible correspondences between observed places, leading to several possible relative pose measurements (tentative correspondences are shown in Fig. 8(b) as black lines); most of these measurements will be outliers and correspond to false matches. We apply the elimination approach of Section V-C to robustly solve the problem,

without explicitly introducing binary variables. Each smart factor in this case only computes the expectation (18). The corresponding estimate is shown in Fig. 8(c), where after the optimization we can distinguish measurements classified as inliers (black lines) from outliers (cyan dashed lines).

VI. Conclusion

A natural structure emerges in several applications in computer vision and robotics: the variables involved in the problem can be partitioned in a set of *target* variables, which are of interest for the user, and a set of *support* variables that are only auxiliary. Moreover, the set of support variable can be partitioned in conditionally independent subsets. We exploit this structure to eliminate each subset of support variables. This elimination naturally leads to the definition of new factors, that we call *smart* factors. This perspective not only allows dramatic reduction in the size of the problem to be solved, but provides natural tools to enhance robustness when performing inference over factor graphs.

Appendix

We prove that nonlinear elimination of $R_2, \dots, R_s$ in (15) leads to the expression (16). We reparametrize (15) in terms of the relative rotations $R_{i,i+1} = R_i^T R_{i+1}$, with $i = 1, \dots, s$:

$$-\log \phi_i^b(A_i^b) = \min_{R_{1,2},\dots,R_{s,s+1}} \sum_{i=1}^{s} \left\| \mathrm{Log}\left(\bar{R}_{i,i+1}^T R_{i,i+1}\right) \right\|^2_{\sigma^2_{i,i+1}}$$
$$\text{subject to} \quad R_{1,2} \dots R_{s,s+1} = R_u^T R_v \tag{19}$$

As in the linear case, to get our smart factor, we solve for the support variables in function of the target variables. Therefore, we compute $\{R_{i,i+1}\}^\star$ from (19) and back-substitute the optimal value in the expression of the factor. This problem has been explored with a different application in [6] and [27]. Therefore we know that the optimal solution can be computed for fixed (unknown) R_u and R_v as:

$$R^\star_{i,i+1} = \bar{R}_{i,i+1} \bar{R}_{>i} \mathrm{Exp}\left(\omega_{i,i+1} \log\left(\bar{R}_{1,s+1}^T R_u^T R_v\right)\right) \bar{R}_{>i}^T$$

where we denote with $\bar{R}_{>i}$ the product $\bar{R}_{i+1,i+2} \dots \bar{R}_{s,s+1}$ (or the identity matrix for $i = s$), $\bar{R}_{1,s+1} = \bar{R}_{1,2} \dots \bar{R}_{s,s+1}$, and $\omega_{i,i+1} = \frac{\sigma^2_{i,i+1}}{\sum_{i=1}^{s} \sigma^2_{i,i+1}}$. As in the linear case, we obtain our smart factor by plugging back the optimal solution $\{R_{i,i+1}\}^\star$ (which is function of R_u to R_v) into (19):

$$\sum_{i=1}^{s} \left\| \mathrm{Log}\left(\bar{R}_{>i} \mathrm{Exp}\left(\omega_{i,i+1} \log\left(\bar{R}_{1,s+1}^T R_u^T R_v\right)\right) \bar{R}_{>i}^T\right) \right\|^2_{\sigma^2_{i,i+1}}$$

where $\{R_{i,i+1}\}^\star$ were computed so to satisfy the constraint $R_{1,2} \dots R_{s,s+1} = R_u^T R_v$ [6]. Noticing that (i) for rotation matrices S_1 and S_2 it holds $\mathrm{Log}\left(S_1 S_2 S_1^T\right) = S_1 \mathrm{Log}(S_2)$, that (ii) the norm is invariant to rotation, (iii) that the Log is the inverse of the exponential map, we prove the claim:

$$-\log \phi_i^b(A_i^b) = \sum_{i=1}^{s} \frac{\sigma^2_{i,i+1}}{\left(\sum_{i=1}^{s} \sigma^2_{i,i+1}\right)^2} \left\| \log\left(\bar{R}_{1,s+1}^T R_u^T R_v\right) \right\|^2 =$$
$$= \left\| \log\left(\bar{R}_{1,s+1}^T R_u^T R_v\right) \right\|^2_{\sum_{i=1}^{s} \sigma^2_{i,i+1}}$$

References

[1] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Intl. J. of Robotics Research*, vol. 31, pp. 217–236, Feb 2012.

[2] K. Konolige, G. Grisetti, R. Kuemmerle, W. Burgard, L. Benson, and R. Vincent, "Efficient sparse pose adjustment for 2D mapping," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, Oct 2010, pp. 22–29.

[3] 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.

[4] L. Polok, V. Ila, M. Solony, P. Smrz, and P. Zemcik, "Incremental block cholesky factorization for nonlinear least squares in robotics," in *Robotics: Science and Systems (RSS)*, 2013.

[5] L. Carlone, R. Aragues, J. Castellanos, and B. Bona, "A linear approximation for graph-based simultaneous localization and mapping," in *Robotics: Science and Systems (RSS)*, 2011.

[6] G. Dubbelman, I. Esteban, and K. Schutte, "Efficient trajectory bending with applications to loop closure," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010, pp. 1–7.

[7] V. Ila, J. M. Porta, and J. Andrade-Cetto, "Information-based compact Pose SLAM," *IEEE Trans. Robotics*, vol. 26, no. 1, 2010, In press. [Online]. Available: http://dx.doi.org/10.1109/TRO.2009.2034435

[8] H. Kretzschmar and C. Stachniss, "Information-theoretic compression of pose graphs for laser-based slam," *Intl. J. of Robotics Research*, vol. 31, no. 11, pp. 1219–1230, 2012.

[9] G. Huang, M. Kaess, and J. Leonard, "Consistent sparsification for graph optimization," in *Proc. of the European Conference on Mobile Robots (ECMR)*, 2012.

[10] N. Carlevaris-Bianco and R. M. Eustice, "Generic factor-based node marginalization and edge sparsification for pose-graph SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2013, pp. 5728–5735, name in key is spelled wrong.

[11] J. Neira and J. Tardos, "Data association in stochastic mapping using the joint compatibility test," *IEEE Trans. Robot. Automat.*, vol. 17, no. 6, pp. 890–897, December 2001.

[12] E. Olson, "Recognizing places using spectrally clustered local matches," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1157–1172, 2009.

[13] N. Sünderhauf and P. Protzel, "Switchable constraints for robust pose graph SLAM," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[14] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," *Intl. J. of Robotics Research*, vol. 32, no. 7, pp. 826–840, 2013.

[15] L. S. C. S. P. Agarwal, G.D. Tipaldi and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2013.

[16] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, February 2001.

[17] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from Internet photo collections," *Intl. J. of Computer Vision*, vol. 80, no. 2, pp. 189–210, November 2008. [Online]. Available: http://phototour.cs.washington.edu/

[18] A. Mourikis and S. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, April 2007, pp. 3565–3572.

[19] M. Li and A. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *Intl. J. of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[20] Y. Jeong, D. Nister, D. Steedly, R. Szeliski, and I. Kweon, "Pushing the envelope of modern methods for bundle adjustment," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 34, no. 8, pp. 1605–1617, 2012.

[21] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Providence, USA, June 2012, pp. 3354–3361.

[22] K. Konolige, "Sparse sparse bundle adjustment," in *British Machine Vision Conf. (BMVC)*, September 2010.

[23] ——, "Large-scale map-making," in *Proc.* 21^{th} *AAAI National Conference on AI*, San Jose, CA, 2004.

[24] J. Folkesson and H. I. Christensen, "Robust SLAM," in *IAV-2004*, Lisboa, PT, July 5-7 2004.

[25] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Trans. Robotics*, vol. 28, no. 1, pp. 61–76, Feb 2012.

[26] T. Minka, "Expectation-Maximization as lower bound maximization," November 1998.

[27] G. Sharp, S. Lee, and D. Wehe, "Multiview registration of 3D scenes by minimizing error between coordinate frames," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, no. 8, pp. 1037–1050, 2004.