2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Modern MAP inference methods for accurate and fast occupancy grid mapping on higher order factor graphs

Vikas Dhiman, Abhijit Kundu, Frank Dellaert and Jason J. Corso

Abstract— **Using the inverse sensor model has been popular in occupancy grid mapping. However, it is widely known that applying the inverse sensor model to mapping requires certain assumptions that are not necessarily true. Even the works that use forward sensor models have relied on methods like expectation maximization or Gibbs sampling which have been succeeded by more effective methods of maximum a posteriori (MAP) inference over graphical models. In this paper, we propose the use of modern MAP inference methods along with the forward sensor model. Our implementation and experimental results demonstrate that these modern inference methods deliver more accurate maps more efficiently than previously used methods.**

I. Introduction

Mobile robot problems like navigation, path planning, localization and collision avoidance require an estimate of the robot's spatial environment; this underlying problem is called robot mapping [2]. Even in environments in which maps are available, the environment may change over time necessitating a mapping ability on the mobile robot. Robot mapping hence remains an active field of research [3]–[5] as it is an important problem in application areas like indoor autonomous navigation, grasping, reconstruction and augmented reality.

Although robot mapping can be performed in many ways—metric or topological; with range sensors, like sonar [6], laser scanners [6] and RGBD [7], or bearing-only sensors [8], [9]—metric mapping with range sensors is the most common. Bearing-only sensors provide estimates up to scale; topological maps still require local metric estimates for certain problems like navigation. We hence focus on metric mapping with range sensors, specifically, laser scanners.

Occupancy grid mapping (OGM) is a popular and useful range-based mapping method [10], [11]. It affords a simple implementation and avoids a need to explicitly seek and match landmarks in the environment [12], [13]. In contrast, it discretizes the environment into cells, squares (2D) or cubes (3D), and associates a random variable with each cell that represents the probability of the cell being occupied or free. Also, unlike surface-based approaches [14], [15], OGM makes it easier to query obstacles, hence collisions, which is critical for applications like robot navigation.

OGM methods vary in how cell occupancy is estimated, but most methods make use of an inverse sensor model that assumes the occupancy of each cell can be estimated independently of the other cells in the map [7], [10], [11], [16]. The main reason for using this independence assumption is computational efficiency. However, the assumption is inaccurate and can lead to overconfident estimates of occupancy in noisy observations [5], [6].

To overcome this limitation, Thrun [6] proposes the use of a forward sensor model and expectation maximization to estimate occupancy. Following this line of work, more recently, Merali et al. [5] defines a Gibbs sampling algorithm based on a conditional estimate of cell occupancy given the rest of the map. Although these methods have relaxed the assumptions of independence, they remain computationally expensive and hence limited in applicability. For example, it is widely known that Gibbs sampling is computationally expensive and can get caught in local maxima [17].

In contrast, in this paper, we explore the use of modern inference algorithms for more effective occupancy grid mapping with forward sensor models. Our contribution in this paper is two fold. Firstly, we introduce the factor graph approach to the occupancy grid mapping problem, which, to the best of our knowledge, has not been applied to this problem. This factor graph formalism makes it plausible to apply modern fast inference algorithms, such as loopy belief propagation [18] and dual decomposition [19].

Secondly, we introduce a class of higher order factors for our factor graph approach. Factor graph inference is exponential in neighborhood size, which requires us to focus on a certain sub-class of factors for tractability, such as the linear constraint-nodes [20] or pattern-based factors [21]. We extend the pattern-based factors, which explicitly compute the potential only for certain factors matching a given set of patterns and otherwise assign a constant. Whereas the pattern-based factors in [21] define each pattern with a fixed value for each node, we generalize these pattern-based factors by allowing for *free* nodes whose value does not impact the computed marginal.

We implement these contributions for effective occupancy grid mapping with a forward sensor model and test our work on both simulated and real data. Our experiments demonstrate the effectiveness of our novel OGM approach, especially, dual decomposition.

Vikas Dhiman[1] and Jason J. Corso[1] were partially supported by FHWA DTFH61-07-H-00023. Jason J. Corso was partially supported by NSF CAREER IIS-0845282 and ARO YIP W911NF-11-1-0090. Frank Dellaert[2] and Abhijit Kundu[2] were partially supported by an ARO MURI W911NF-11-1-0046. We thank Stan Birchfield and Brian Peasley for discussions and early efforts in this work.

Vikas Dhiman[1] and Jason J. Corso[1] are with Department of Computer Science and Engineering, SUNY at Buffalo, NY, USA {vikasdhi,jcorso}@buffalo.edu

Abhijit Kundu[2] and Frank Dellaert[2] are with College of Computing, Georgia Tech, GA, USA {abhijit.kundu,frank}@gatech.edu

We thank C. Stachniss for providing *albert-b-laser* dataset [1].

II. Background and Related work

A main contribution of our paper is the application of modern MAP algorithms to occupancy grid mapping. Although there are many MAP inference algorithms [22] that work well for various problems, in this paper, we focus on belief propagation [18] and dual decomposition [19] mainly because of their ability to handle higher order factors.

Belief propagation (BP) [23] was introduced as an algorithm to compute marginals over trees. Surprisingly, it was found to work well on graphs with loops. Later it was found that the convergent solution to belief propagation corresponds to the minima of the so-called Bethe free energy [24], which not only provided a theoretical justification for application of belief propagation to graphs with loops, but also solves the convergence problem by providing an objective function which can be minimized directly. Later, Fractional BP [25] was introduced. Inspired by the Bethe free energy formulation of BP, it suggested using a better free energy approximation by scaling the terms appropriately in the message update equation.

In an independent work, Wainwright et al. [26] introduce Tree Re-Weighted (TRW) message passing algorithm that uses re-weighting of edges and messages similar to Fractional BP. They also formulate the MAP estimation problem as a linear program over the so-called marginal polytope. The Langrangian dual of this LP problem is convex and provides the upper bound to the original problem. The family of algorithms that optimize the Lagrangian dual of the original combinatorial problem are called dual decomposition (DD). The dual of the problem can be decomposed in different ways, for example, as set of spanning trees in TRW [26] or one problem per factor [19]. Kolmogorov et al. [18] improved over the work of [26] to introduce a convergent algorithm called TRW-S. More recently, accelerated dual decomposition [27] was introduced that provably converges the upper bound faster than earlier approaches by smoothing the Lagrangian dual of the problem.

Another contribution of our paper is how we perform efficient inference with higher order factors (or potentials). Many researchers approach this problem by considering a class of functions for which higher order factors can used efficiently, for example, Potetz et al. introduce a class of potentials called linear constraint nodes [20] and Komodakis et al. approach pattern-based class of potentials [21]. Another approach to handle higher order potentials [28] is to adaptively restrict the sample space of nodes by using initial estimates. Our work proposes a generalization of the pattern-based potentials of [21] allowing for *free* nodes to appear within a pattern.

III. Problem definition

Consider a robot—equipped with a laser scanner—moving in a static environment, and assume the position of the robot associated with each laser measurement is given. Our task is to estimate occupied regions and free regions, so that the robot can avoid collisions with occupied regions and plan its movement in free regions. We divide the area to be mapped into N discrete cells. Let x_i denote the state of cell i, which can take values from label set $L_i = \{0, 1\}$, where 0 (resp. 1) denotes that the cell is free (resp. occupied). For convenience, we denote the full map (the state for all N cells) as $\mathbf{x} = [x_i]_{1 \le i \le N}^\top$ taking values from sample space $\Omega = \prod_{1 \le i \le N} L_i$.

Let z_f denote the f^{th} laser range measurement when captured from (known) pose g_f. The problem is to find the probability of all cells of in the map being occupied given all t observations, $\mathbf{z} = [z_f]_{1 \le f \le t}^\top$ and $\mathbf{g} = [g_f]_{1 \le f \le t}^\top$:

$$p(x_i = 1 | \mathbf{z}, \mathbf{g}) = \sum_{\mathbf{x} \in \Omega : x_i = 1} p(\mathbf{x} | \mathbf{z}, \mathbf{g}) \quad \forall 1 \le i \le N \ . \quad (1)$$

Alternatively, we can focus on the maximum posterior map:

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in \Omega} p(\mathbf{x} | \mathbf{z}, \mathbf{g}). \quad (2)$$

Problems (1) and (2) are related but yield different results. While (1) is useful to keep track of uncertainty in an incremental fashion, (2) provides a more meaningful result in the joint occupancy configuration maximizing posterior probability. Clearly, a naïve solution to either problem would have complexity that is exponential in the number of cells. We hence focus on an approximate solution to this problem.

A. Mapping with an inverse sensor model

Commonly used occupancy grid mapping algorithms [7], [10], [11] make the simplifying assumption that each grid cell is independent of all other map cells:

$$p(\mathbf{x} | \mathbf{z}, \mathbf{g}) = \prod_{1 \le i \le N} p(x_i | \mathbf{z}, \mathbf{g}) \ . \quad (3)$$

The probability of each cell can be easily computed independent of each other, by a simple Bayes formulation:

$$p(x_i | \mathbf{z}, \mathbf{g}) = \frac{p(z_t, g_t | x_i, \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1}) p(x_i | \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1})}{p(z_t, g_t | \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1})} \ . \quad (4)$$

Assuming a static world, $p(z_t, g_t | x_i, \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1}) = p(z_t, g_t | x_i)$, as is commonly done, the above equation (4) can be simplified [5] to:

$$p(x_i | \mathbf{z}, \mathbf{g}) = \frac{1}{Z'} \frac{p(x_i | z_t, g_t)}{p(x_i)} p(x_i | \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1}) \quad (5)$$

$$= \frac{1}{Z' p^{t-1}(x_i)} \prod_{1 \le f \le t} p(x_i | z_f, g_f) \ , \quad (6)$$

where Z' is a normalizing factor that is independent of x_i, $p(x_i)$ is the prior probability for cell x_i and $p(x_i | z_f, g_f)$ is called the *inverse sensor model*.

B. Mapping with a forward sensor model

However, the independent cell assumption is inaccurate and can lead to overconfident estimates of occupancy in noisy observations [5], [6]. In the absence of the independent cell assumption, we can still factorize the posterior probability in terms of a forward sensor model.

In this formulation, we make two assumptions, a) static world assumption, $p(z_t | \mathbf{x}, \mathbf{g}) = p(z_t | \mathbf{x}, g_t)$ and b) pose-map

independence, $p(g_t|\mathbf{x}, \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1}) = p(g_t|\mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1})$. With these assumptions, the posterior evaluates [5] to

$$p(\mathbf{x}|\mathbf{z}, \mathbf{g}) = \frac{1}{Z} p(z_t|\mathbf{x}, g_t) p(\mathbf{x}|\mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1}) \tag{7}$$

$$= \frac{1}{Z} p(\mathbf{x}) \prod_{1 \leq f \leq t} p(z_f|\mathbf{x}, g_f) \ , \tag{8}$$

where Z is a normalizing constant independent of $\mathbf{x}$ and $p(\mathbf{x})$ is the prior. For the rest of the paper we assume no prior information about the maps, hence the prior is a constant and included in the normalizing constant Z.

The problem of estimating the occupancy map is intractable with the above formulation, as it still depends on the entire map, which has an exponential sample space. However, we can make use of the fact that a laser measurement f depends only on a small portion of the map $\mathbf{x}_f \subseteq \mathbf{x}$, hence simplifying the formulation to:

$$p(\mathbf{x}|\mathbf{z}, \mathbf{g}) = \frac{1}{Z} \prod_{1 \leq f \leq t} p(z_f|\mathbf{x}_f, g_f) \ . \tag{9}$$

The term $p(z_f|\mathbf{x}_f, p_f)$ is called the *forward sensor model*. With this formulation, we are in a position to describe the problem as *factor graph*. The above simplification is necessary to keep the factor graph sparsely connected and hence tractable.

C. Representation as a factor graph

The occupancy grid mapping problem can be expressed as energy minimization over a factor graph. Let all cells in the map be the variable nodes V and all the laser measurements be factor nodes F. There exists an undirected edge (i, f), if and only if the laser range measurement $p(z_f|\mathbf{x}_f, g_f)$ depends on the cell occupancy x_i. In this paper, we assume that each laser range measurement depends on only those cells that the laser passes through for given pose g_f. Let E be set of all such edges:

$$E = \{(i, f) : i \in V, f \in F, \text{laser } f \text{ passes through cell } i\} \ . \tag{10}$$

The bipartite graph $G = (V, F, E)$ represents the structure of factorization in (9) and is hence called a factor graph [18]. Note that neighborhood $n(i)$ of any variable node i only consists of the factors nodes, $n(i) \subseteq F \, \forall i \in V$ and vice versa. Fig. 1 shows the factor graph diagrammatically. Note that observed nodes form the part of factor and are not part of the factor graph.

In terms of factor graph G the problems (1) and (2) along with factorization obtained in (9) can be written as:

$$P_i(x_i = l_i) = \sum_{\mathbf{x} \in \Omega : x_i = l_i} P(\mathbf{x}) \qquad \forall i \in V \tag{11}$$

$$\mathbf{x}* = \arg\max_{\mathbf{x} \in \Omega} P(\mathbf{x}) \tag{12}$$

$$P(\mathbf{x}) = \frac{1}{Z} \prod_{f \in F} P_f(\mathbf{x}_f) \ , \tag{13}$$

where $l_i \in L_i$ denotes an element from the label set L_i of node $i \in V$. By representation of occupancy grid mapping as factor graph the following equivalence holds true, $P_i(x_i = l_i) \equiv p(x_i = l_i|\mathbf{z}, \mathbf{g})$, $P(\mathbf{x}) \equiv p(\mathbf{x}|\mathbf{z}, \mathbf{g})$ and $P_f(\mathbf{x}_f) \equiv p(z_f|\mathbf{x}_f, g_f)$.

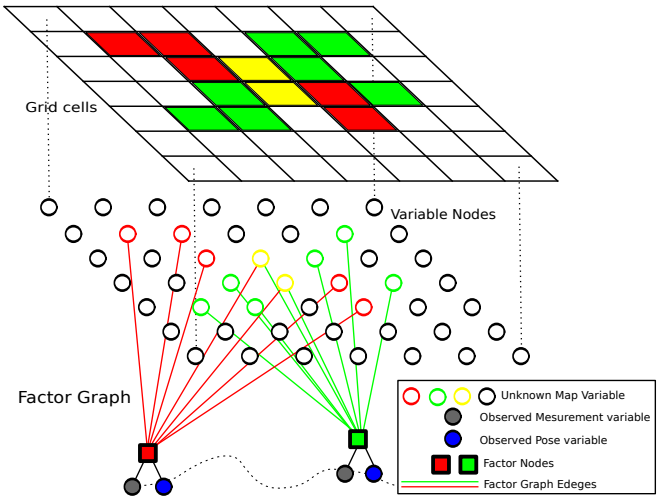

Fig. 1. Representation of occupancy grid mapping as factor graph

IV. Markov Chain Monte Carlo methods (Metropolis Hastings)

We implement and evaluate a generalization of Merali's [5] Gibbs sampling algorithm for estimating maps. Metropolis Hastings is another popular MCMC algorithm used for sampling from complex probability distributions. Gibbs sampling can be shown to be a specialization of Metropolis Hastings [29].

Metropolis Hastings requires a *transition probability* $Q(\mathbf{x}'|\mathbf{x}^r)$, that depends on current sample $\mathbf{x}^r$ and guides the random walk in the high-dimensional space. We randomly sample a point $\mathbf{x}'$ from from $Q(.)$ and it is either accepted or rejected based on the *acceptance probability* a:

$$a = \frac{P(\mathbf{x}')Q(\mathbf{x}^r|\mathbf{x}')}{P(\mathbf{x}^r)Q(\mathbf{x}'|\mathbf{x}^r)} \ . \tag{14}$$

If $a \geq 1$, then the new point $\mathbf{x}'$ is accepted otherwise it is accepted with probability a. Here acceptance means that the point in the next iteration is taken as the sampled point otherwise the earlier point is retained. The interested reader is referred to [17], [29] for further details about Metropolis Hastings.

For our experiments, we choose a symmetric uniform transition probability. We uniformly sample a cell from the map and flip the state of the sampled cell to get the proposal point $\mathbf{x}'$. This is equivalent to sampling from the probability density:

$$Q(\mathbf{x}'|\mathbf{x}^r) = \begin{cases} \frac{1}{N} & \text{if } \|\mathbf{x}' - \mathbf{x}^r\|_1 = 1 \\ 0 & \text{otherwise} \end{cases} \ , \tag{15}$$

where N is the number of cells in the map and $\|.\|_1$ is the L1-norm. The first case in (15) enforces that only one cell (or dimension) in the map can change its state. Since this is a symmetric transition probability, the acceptance ratio is just

the ratio of target probabilities. Also note that the ratio of probability distributions can be efficiently computed because of the factorization obtained in (13):

$$a = \frac{P(\mathbf{x}')}{P(\mathbf{x}^r)} = \frac{\prod_{f \in n(i)} P_f(\mathbf{x}'_f)}{\prod_{f \in n(i)} P_f(\mathbf{x}^r_f)} \ , \tag{16}$$

where i is the (sampled) cell whose state is different in $\mathbf{x}'$ and $\mathbf{x}^r$, and $n(.)$ denotes the neighborhood of a vertex in graph G. The above simplification uses the fact that only those terms that depend on the state of i^{th} cell need to be computed. By definition of graph G, only the neighboring factors $f \in n(i)$ depend on the state of cell i.

It is in general difficult to detect when the sampling algorithm has converged. In practice, we often run the sampling algorithm for a fixed number of iterations. Alg. 1 lists the pseudo code for Metropolis Hastings algorithm.

Algorithm 1: Metropolis Hastings

Data: Factor Graph $G = (V, F, E)$;
Maximum number of iterations N;
Result: $\mathbf{x}^r$
Initialize the map $\mathbf{x}^0$ randomly.;
$r = 0$;
while $r < N$ **do**
  Randomly choose a cell $i : i \in V$;
  Flip its state $x'_i = \neg x^r_i$ in $\mathbf{x}^r$ to get $\mathbf{x}'$;
  Compute acceptance probability a by (16);
  Sample random number $q : 0 \leq q \leq 1$;
  if $a \geq 1$ *or* $a \geq q$ **then**
    Accept proposed point, $\mathbf{x}^{r+1} = \mathbf{x}'$;
  else
    Reject proposed point, $\mathbf{x}^{r+1} = \mathbf{x}^r$;
  $r \leftarrow r + 1$;

A. Heat map

As we uniformly sample cells from the map, we notice that not all cells are equally important in mapping. There are three kinds of regions in an occupancy map: occupied, free and unexplored. Sampling and analyzing a cell in an unexplored region is not very helpful as we do not have any evidence for the region. On the other hand, the central regions of free areas are not very interesting as all factors usually agree on their state. The uncertainty tends to lie along the boundaries of free and occupied regions. This is the region we want to focus on.

We hence employ a *heat map* to bias our sampling along the boundaries of free and occupied regions. We maintain a vector of cells $\mathbf{x}_h$ that form the "interesting" region of the map. In our experiments, we take the last cell spanned by each laser measurement as an "interesting" cell and add it to the *heat map*, $\mathbf{x}_h$. We use a sampling bias of $1 : 4$ for cells outside the heat map to cells within the heat map. We compare both Metropolis Hastings with and without the heat map in our experiments.

V. Modern inference algorithms

As discussed in Sec. II, last decade gave rise to faster and more accurate MAP inference algorithms [22]. Because of their ability to handle higher order factors [20], [21], we explore belief propagation and dual decomposition in the problem of occupancy grid mapping.

A. Belief Propagation

The sum product algorithm over factor graphs [18] is a powerful yet simple algorithm to compute marginals of expression, of the form (1), that can be decomposed into factors of the form (9). The algorithm provides exact marginals in the case when the graphs have no loops. For graphs with loops the algorithm has been shown to converge in most of practical problems.

The sum product algorithm works by sending messages along the edges of the factor graph. The messages can be understood as the *beliefs* of the source node about destination states. Mathematically, these beliefs are the probabilities of the destination states marginalized over the neighbours of the source except the destination itself. These messages are defined on a directed edge, with a different message value for each state of the variable node involved.

Let $\mu^r_{f \to i}(l_i)$ represent the message from node $i \in V$ to node $f \in F$ for state $x_i = l_i$ at any iteration r of the algorithm. With a similar convention we take $\mu^r_{i \to f}(l_i)$ to denote an update in the opposite direction. We use the following equations to update the messages on an edge depending on whether the direction of the edge is from variable node to factor node or vice versa:

$$\mu^{r+1}_{f \to i}(l_i) = \sum_{\mathbf{x}_f \in \Omega_f : x_i = l_i} P_f(\mathbf{x}_f) \prod_{j \in n(f) \setminus i} \mu^r_{j \to f}(x_j) \tag{17}$$

$$\mu^{r+1}_{i \to f}(l_i) = \prod_{h \in n(i) \setminus f} \mu^r_{h \to i}(l_i) \ , \tag{18}$$

where $\Omega_f = \prod_{i \in n(f)} L_i$ denotes the sample space of the neighborhood of factor f in graph G. On convergence, the belief of variable nodes can be computed by the product of incoming messages:

$$P(x_i = l_i) = \prod_{f \in n(i)} \mu^r_{f \to i}(l_i) \ . \tag{19}$$

This is called the sum product belief propagation (BP) algorithm.

One can compute the maximizing assignment instead of marginals by computing the max product instead of sum product in (17) and finally choosing the maximizing assignment of incoming messages:

$$\mu^{r+1}_{f \to i}(l_i) = \max_{\mathbf{x}_f \in \Omega_f : x_i = l_i} P_f(\mathbf{x}_f) \prod_{j \in n(f) \setminus i} \mu^r_{j \to f}(x_j) \tag{20}$$

$$x^*_i = \arg\max_{x_i \in L_i} \prod_{f \in n(i)} \mu^r_{f \to i}(l_i). \ . \tag{21}$$

This form of the algorithm is called max product BP.

Belief propagation was initially designed to work on factor graphs without loops. In such a case one can start message

updates from the leaf nodes and a node can be "triggered" to pass on the messages when messages from all but one neighbors are available. However, for graphs with loops various update sequences have been suggested that vary from problem to problem. For example, in vision problems, where the factor graph is a 2D grid, horizontal and then vertical sweeps have been shown to produce good results. For our implementation, we choose random update sequence, i.e., a random edge is selected from the graph for each iteration of message update.

B. Subgradient Dual decomposition

The dual decomposition algorithm employs the theory of Lagrangian duals to find a convex upper bound of the original combinatorial optimization problem. Here we explain the implementation of the algorithm without going into mathematical proofs. The interested reader is referred to [19], [21], [27] for proofs and more variations of the algorithm.

The underlying idea for dual decomposition is to split the maximization problem into *slave* problems that can be efficiently maximized. In a factor graph formulation the natural slave problem is one corresponding to each factor:

$$\mathbf{x}^f = \arg\max_{\mathbf{x}^f} P_f(\mathbf{x}^f) \prod_{i \in n(f)} \exp\left(-\mu_{if}(x_i^f)\right) \ , \qquad (22)$$

where $\mathbf{x}^f = \{x_i^f\}_{i \in n(f)}$ is the optimum assignment as determined by the corresponding slave problem. And $\mu_{if}(x_i^f)$ is the message (also the Lagrangian multiplier) from node i to f about state x_i^f. The above slave problem is usually written in the form of negative log likelihood:

$$\mathbf{x}^f = \arg\min_{\mathbf{x}^f} \theta_f(\mathbf{x}^f) + \sum_{i \in n(f)} \mu_{if}(x_i^f) \ , \qquad (23)$$

where $\theta_f(\mathbf{x}^f) = -\log P_f(\mathbf{x}^f)$ is the negative log likelihood corresponding to the factor.

In each iteration of the algorithm all slave problems are allowed to choose their optimum assignment independently. If all the factors agree on the assignments, then we have reached the global optimum. Often this is not the case. In case of disagreement, we decrease the belief of all the disagreeing slave problems about their respective optimums by sending appropriate messages. It can be shown that as long as we can increment the messages by decreasing step size in each iteration, the algorithm is guaranteed to converge to an approximate solution of the original problem [19].

Pseudocode for dual decomposition (DD) is provided in Alg. 2. Apart from input factor graph $G = (V, F, E)$ and label set $\{L_i\}_{i \in V}$ introduced in Sec III-C, dual decomposition depends on a step size α. We note that step size is an important attribute and affects the speed of the algorithm. For illustration, we show the convergence of dual decomposition with different step sizes on *cave* dataset in Fig. 2. Note that erring on the higher side causes oscillations, while erring on the lower side can cause convergence to be too slow.

Upon convergence or completing a maximum number of iterations, we can compute the optimal assignment for

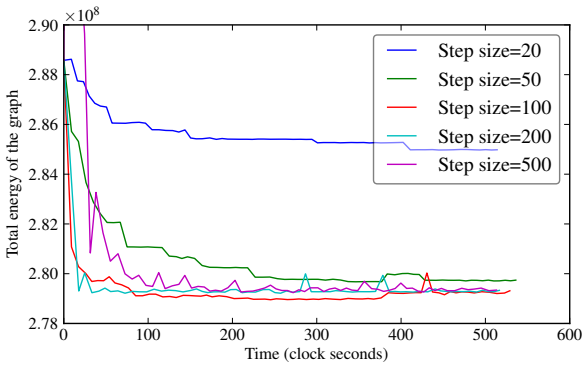

Fig. 2. The rate of convergence in subgradient dual decomposition depends on step size

Algorithm 2: Subgradient Dual Decomposition

Data:
Factor Graph $G = (V, F, E)$
Step size $\alpha > 0$
Maximum number of iterations N
Result: Labels $\{x_i^f\}_{(i,f) \in E}$, Messages $\{\mu_{if}(x_i)\}$
$\mu_{if}(x_i) \leftarrow 0 \qquad \forall (i, f) \in E, x_i \in L_i$
$r \leftarrow 1$
while $r < N$ **do**
  for $f \in F$ **do**
    $\mathbf{x}^f \leftarrow \arg\min_{\mathbf{x}^f} \left(\theta_f(\mathbf{x}^f) + \sum_{i \in n(f)} \mu_{if}(x_i^f) \right)$
  `// For disagreeing nodes`
  for $i \in V : \exists f, f' \in n(i) : x_i^{f'} \neq x_i^f$ **do**
    for $f \in n(i)$ **do**
      $\mu_{if}(x_i^f) \leftarrow \mu_{if}(x_i^f) + \frac{\alpha}{r}$
  $r \leftarrow r + 1$

variable nodes with disagreeing slaves from the messages:

$$x_i = \arg\max_{x_i \in L_i} \sum_{f \in n(i)} \mu_{if}(x_i). \qquad (24)$$

Note that the above equation is only valid for disagreeing slaves. When the slaves agree, we can simply pick the agreed upon value.

Dual decomposition is an optimization algorithm; hence it only solves the MAP problem (12), but not the marginal problem (11), which may be considered a limitation.

VI. Higher order factors and efficiency

The message update equation (17) in the BP algorithm and the slave minimization (23) in the DD algorithm are, in general, exponential in the size of neighborhood of factor f, which is computationally expensive for higher order factor graphs. This motivates us to seek a generic class of factors that can be efficiently applied to the belief propagation (BP) and dual decomposition (DD) algorithms. We begin by introducing common forward sensor models that need to

fit our class of factors followed by their generalization and then their application to the BP and DD algorithms.

A. Forward sensor models

Forward sensor models estimate the sensor reading given the environment. In this section, we describe two commonly used sensor models a) Gaussian and b) piecewise constant sensor models.

1) Gaussian sensor model (GSM): Assuming Gaussian noise σ, the range measurement by a laser sensor is given by:

$$p(z_f|\mathbf{x}_f, g_f) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\bar{z}_f(\mathbf{x}_f, g_f) - z_f)^2}{2\sigma^2}\right), \quad (25)$$

where $\bar{z}_f(\mathbf{x}_f, g_f)$ is the distance of first occupied cell in $\mathbf{x}_f$ starting from pose g_f.

2) Piecewise constant sensor model (PCSM): It is common in graphical models to have simple factors that assign high probability to expected configurations and low probability to all other states. Hence, we propose the following factor, which we use in all our experiments,

$$p(z_f|\mathbf{x}_f, g_f) = \frac{1}{Z}\begin{cases} 1 & \text{if } \mathbf{x}_f = \mathbf{R}_{f1} \\ \exp(-900) & \text{if } \mathbf{x}_f = \mathbf{R}_{f2} \\ \exp(-1000) & \text{otherwise} \end{cases}, \quad (26)$$

where Z is the normalization constant, $\mathbf{R}_{f1} = [0, 0 \dots 0, 1]^\top$ denotes the all-free-but-last-occupied cell pattern and $\mathbf{R}_{f2} = [0, 0 \dots 0, 0]^\top$ denotes all free cells. The second case indicates that we are more averse to estimating the reflecting cell closer to the robot as compared to estimating it away from the robot.

B. Generalization to pattern-based factors

We define a class of *pattern-based* higher order factors that are a generalization of those discussed by Komodakis et al. [21]. These are factors of the form:

$$p(z_f|\mathbf{x}_f, g_f) = \begin{cases} \psi_m & \text{if } \mathbf{x}_f \sim \mathbf{R}_m \ \ \forall 1 \le m \le M \\ \psi_{\text{min}} & \text{otherwise} \end{cases}, \quad (27)$$

where $\mathbf{R}_m$ is one of the mutually exclusive M patterns and expression $\mathbf{x}_f \sim \mathbf{R}_m$ denotes that vector $\mathbf{x}_f$ "matches" pattern $\mathbf{R}_m$. A pattern, $\mathbf{R}_m = (n_0^m(f), \mathbf{r}^m)$, is defined by a non-empty set of *fixed* nodes $n_0^m(f) \subseteq n(f)$ that are expected to have desired values $\mathbf{r}^m$, while the state of remaining *free* nodes can take any value from the label set. A configuration $\mathbf{x}_f$ "matches" pattern $\mathbf{R}_m$ if the state of *fixed* nodes is the same as the desired values, $x_i = r_i^m \forall i \in n_0^m(f)$.

It is clear that PCSM (26) is an instance of pattern-based factors. It is also possible to represent GSM (25) as a pattern-based factor. We use the fact that the term $\bar{z}_f(\mathbf{x}_f, g_f)$, just depends on the first occupied cell. We define pattern $\mathbf{R}_m$ such that the first occupied cell is the m^{th} cell in $n(f)$:

$$n_0^m(f) = n(f)_{1:m} \quad (28)$$

$$\mathbf{r}^m = \{r_i^m = 0\}_{i \in n(f)_{1:m-1}} \text{ and } r^m_{n(f)_m} = 1 \quad (29)$$

$$R_m = (n_0^m, \mathbf{r}^m) \ , \quad (30)$$

where $n(f)_k$ is the k^{th} cell that is traced by the laser. With this formulation, we will have $M = n(f)$ patterns to describe the GSM in the form (27) with ψ_m given by

$$\psi_m = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\delta(m) - z_f)^2}{2\sigma^2}\right) \ , \quad (31)$$

where $\delta(m)$ is the distance of m^{th} cell from the robot starting point g_f. Also note that in this formulation the only case for *otherwise* is the one when all cells are 0 (or free).

C. Efficient sum product

To efficiently compute (17) for the pattern-based factors defined we need the messages to be normalized. We assume that the messages $\mu^r_{i\to f}(l_i)$ are normalized over the label set L_i to sum up to one, $\sum_{l_i \in L_i} \mu^r_{i\to f}(l_i) = 1$. Using the messages as a probability measure for the state of source nodes, we can compute the probability of each pattern being true:

$$p(\mathbf{x_f} \sim \mathbf{R}_m | x_i = l_i) = \begin{cases} 0 & \text{if } i \in n_0^m(f) \text{ and } l_i \neq r_i^m \\ \prod\limits_{j \in n_0^m(f) \setminus i} \mu^r_{j\to f}(r_j^m) & \text{otherwise} \end{cases}. \quad (32)$$

The message update equation (17) can be written in terms of probability of patterns:

$$\mu^{r+1}_{f\to i}(l_i) = \sum_{m \le M} \psi_m p(\mathbf{x_f} \sim \mathbf{R}_m | x_i = l_i) + \psi_{\text{min}} p_{\text{otherwise}} \ , \quad (33)$$

where $p_{\text{otherwise}} = 1 - \sum_{m \le M} p(\mathbf{x_f} \sim \mathbf{R}_m | x_i = l_i)$.

The message update can be computed in $O(M|n(f)|)$ using (33), instead of $O(L_i^{|n(f)|})$ in the general case. Note that for GSM the number of patterns is same as the size of neighborhood, while for PCSM the number is fixed. Hence, the message update step is quadratic in neighborhood size for GSM and linear for PCSM.

D. Efficient dual decomposition

We note that slave function (23) is composed of two terms: the factor itself and the messages. While minimizing for each pattern we can make use of the fact that the value of θ_f is constant for a pattern and hence we need to focus only on minimizing the messages. Minimizing the messages is trivial as each term can be minimized independently.

While minimizing the *otherwise* case of patterns, we must ensure the exclusivity from explicit patterns. If the space for the otherwise case is small (like in the Gaussian sensor model), we simply go over the entire space to find the minimum value. Otherwise, we keep finding the n-best minimizations of messages until we get a minimization that does not match any of the patterns already considered. For example, for the piecewise constant sensor model, there are only two patterns that need to be checked for exclusion. Hence, the otherwise term can be easily minimized by finding the three best assignments that minimize the messages

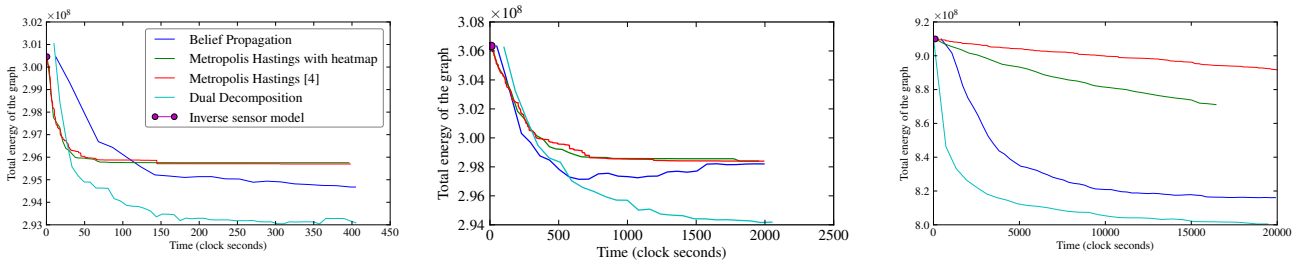

Fig. 3. Comparison of convergence rate of different algorithms on occupancy grid mapping. From left to right, convergence rate on three datasets is shown: 1) *cave*, 2) *hospital section* and 3) *albert-b* [1] dataset. While sampling methods like Metropolis hastings converge quickly they stay far from optimum energy. On the other hand modern inference algorithms like belief propagation and dual decomposition reach closer to an optimum value. The legends are listed only once for clarity.

term. Note that message update for max product BP (20), can be done by following exactly the same logic.

VII. Experiments

We run experiments on simulated as well real data. The simulated data is generated using *Player/Stage* [30] project. We use multiple map bitmaps bundled along with player/stage library. The robot motion is generated using the wander driver. The robot is allowed to wander in the map for 2 minutes aggregating approximately 270,000 laser measurements.

For real data, we have used the *albert-b-laser* dataset provided by C. Stachniss from University of Freiburg. The dataset was captured by a B21r robot with a SICK PLS moving through a lab at University of Freiburg. This data set was obtained from the Robotics Data Set Repository (Radish) [1].

To evaluate the convergence rate of each algorithm, we plot total energy (negative log likelihood) of the graph with respect to CPU ticks used by the algorithm. The plots of energy convergence with respect to time for cave dataset is shown in Fig. 3. This data clearly show the improvement from moving to belief propagation and dual decomposition, which, in all cases, leads to lower energies faster than our baselines. DD outperforms BP in typical cases.

In all our experiments we do not use any occupancy prior, although Merali et al. [5] suggest using an occupancy prior of 0.3 for better convergence. We use a step size of 50 for dual decomposition and piecewise constant sensor model. Also, we prefer piecewise constant sensor model over Gaussian sensor model because of the former being faster which is a consequence of having fewer patterns in the pattern-based factor formulation. We have implemented the algorithms in C++ and the code is available at the authors' websites.

A. Discussion

As is evident from the convergence comparison in Fig. 3, sampling algorithms (Metropolis Hastings with/without heatmap) are liable to getting stuck in a local minima. This is also an artifact of the simple transition probability where we flip only one cell at a time. Even from the qualitative results for sampling algorithms (Fig. 4), we see that the walls are thinner than the corresponding results in other algorithms which shows the inability of sampling-based algorithms to form lower energy and thicker walls for the piecewise constant sensor model. The downside of being biased towards thinner walls is evident in the *albert-b* dataset (see Fig 4), as we get ragged walls for the sampling algorithms.

VIII. Conclusion and Future Work

Dual decomposition is faster because it focuses on disagreeing nodes. However, step size is a crucial parameter that affects the speed of convergence. On the other hand, sum product belief propagation does not depend on any parameter, but has no preference for the disagreeing nodes. This combination obviously hints towards an algorithm where we perform belief propagation over disagreeing nodes only. The state of art variations of these algorithms, like Sequential Tree Re-Weighted (TRW-S) belief propagation [31], accelerated dual decomposition [27], are steps in this direction. Also, a recent comparative study [22] points towards other candidate methods, e.g. polyhedra based methods, that outperform than the dual decomposition class of methods. Even without using these more recent algorithmic developments, we get stronger performance than methods used so far. This only serves to prove our assertion that modern inference methods should be used for occupancy grid mapping.

References

[1] A. Howard and N. Roy, "The robotics data set repository (radish)," 2003. [Online]. Available: http://radish.sourceforge.net/
[2] S. Thrun, "Robotic mapping: A survey," *Exploring Artificial Intelligence in the New Millenium*, 2002.
[3] D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Occupancy grid models for robot mapping in changing environments." in *AAAI*, 2012.
[4] K. Nagla, M. Uddin, and D. Singh, "Improved occupancy grid mapping in specular environment," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1245 – 1252, 2012.
[5] R. Merali and T. Barfoot, "Occpancy grid mapping with markov chain monte carlo gibbs sampling," in *ICRA*, 2013.
[6] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous Robots*, vol. 15, no. 2, pp. 111–127, 2003.
[7] R. Newcombe, A. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *ISMAR*, 2011.
[8] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 6, pp. 1052–1067, 2007.
[9] A. Kundu, K. Krishna, and C. Jawahar, "Realtime multibody visual SLAM with a smoothly moving monocular camera," in *ICCV*, 2011.

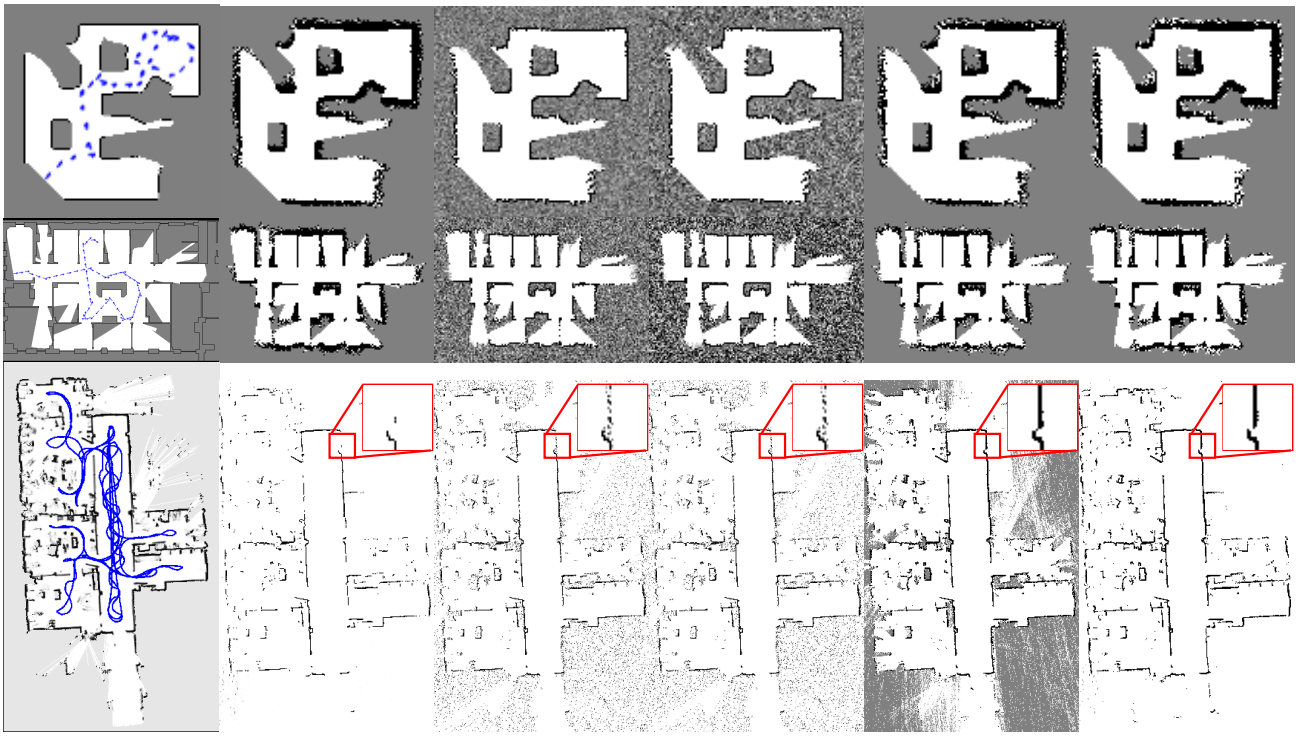

Fig. 4. Qualitative results on different datasets. Each row represents a different dataset while each column represents a different algorithm. The columns correspond to the following algorithms (from left to right): 1) Ground truth with the trajectory of the robot 2) Inverse sensor model 3) Metropolis Hastings without heat map 4) Metropolis Hastings with heat map 5) Belief Propagation (BP) 6) Dual decomposition (DD). The rows correspond to the following datasets (from top to bottom): 1) cave 2) hospital section 3) albert-b. The grainy-ness in columns (3) and (4) is an artifact of sampling algorithms, when we sample over finite number of sample to compute expected state of a cell. Also note the missing or ragged walls in first 3 algorithms, while BP and DD are able to converge to thick solid walls. [1].

[10] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[11] H. P. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI magazine*, vol. 9, no. 2, p. 61, 1988.

[12] K. Sugihara, "Some location problems for robot navigation using a single camera," *Computer Vision, Graphics, and Image Processing*, vol. 42, no. 1, pp. 112 – 129, 1988.

[13] M. Betke and L. Gurvits, "Mobile robot localization using landmarks," *Robotics and Automation, IEEE Transactions on*, vol. 13, no. 2, pp. 251–263, 1997.

[14] M. Ruhnke, R. Kummerle, G. Grisetti, and W. Burgard, "Highly accurate maximum likelihood laser mapping by jointly optimizing laser points and robot poses," in *ICRA*. IEEE, 2011, pp. 2812–2817.

[15] J. Ryde, V. Dhiman, and R. Platt, "Voxel planes: Rapid visualization and meshification of point cloud ensembles," in *IROS*, November 2013.

[16] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *ICRA*, vol. 2, 1985, pp. 116–121.

[17] J. S. Liu, *Monte Carlo Strategies in Scientific Computing*. Springer, 2002.

[18] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 498–519, 2001.

[19] D. Sontag, A. Globerson, and T. Jaakkola, "Introduction to dual decomposition for inference," *Optimization for Machine Learning*, vol. 1, 2011.

[20] B. Potetz, "Efficient belief propagation for vision using linear constraint nodes," in *CVPR*, 2007.

[21] N. Komodakis and N. Paragios, "Beyond pairwise energies: Efficient optimization for higher-order MRFs," in *CVPR*, 2009.

[22] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, J. Lellmann, N. Komodakis, *et al.*, "A comparative study of modern inference techniques for discrete energy minimization problems," in *CVPR*, 2013.

[23] J. Pearl, "Fusion, propagation, and structuring in belief networks," *Artificial Intelligence*, vol. 29, no. 3, pp. 241 – 288, 1986.

[24] J. S. Yedidia, W. T. Freeman, Y. Weiss, *et al.*, "Generalized belief propagation," in *NIPS*, 2000.

[25] W. Wiegerinck, T. Heskes, *et al.*, "Fractional belief propagation," in *NIPS*, 2003.

[26] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, "MAP estimation via agreement on trees: message-passing and linear programming," *Information Theory, IEEE Transactions on*, vol. 51, no. 11, pp. 3697–3717, 2005.

[27] V. Jojic, S. Gould, and D. Koller, "Accelerated dual decomposition for MAP inference," in *ICML*, 2010.

[28] X. Lan, S. Roth, D. Huttenlocher, and M. J. Black, "Efficient belief propagation with learned higher-order markov random fields," in *ECCV*, 2006.

[29] D. J. MacKay, "Introduction to monte carlo methods," in *Learning in graphical models*. Springer, 1998, pp. 175–204.

[30] B. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the 11th International Conference on Advanced Robotics*, 2003.

[31] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 10, pp. 1568–1583, 2006.