Handling Local and Global Ambiguities via a Generalized Graph SLAM Framework based on Multimodal and Hyperedge Constraints

Max Pfingsthorn and Andreas Birk

***Abstract*— Graph-based Simultaneous Localization and Mapping (SLAM) has experienced a recent surge towards robust methods. These methods take the combinatorial aspect of data association into account by allowing decisions of the graph topology to be made during optimization. In this paper, the Generalized Graph SLAM framework for SLAM under ambiguous data association is presented, and a formal description of using hyperedges to encode uncertain loop closures is given for the first time. The framework combines both hyperedges and multimodal Mixture of Gaussian constraints to cover all sources of ambiguity in SLAM. An extension of the authors' multimodal *Prefilter* method is developed to find good initial conditions in such a generalized multimodal hypergraph. Experiments on synthetic datasets show that the novel multimodal hypergraph *Prefilter* method is both significantly more robust and faster than other robust state-of-the-art methods.**

I. Introduction

Today's robotics research pushes the envelope on where solutions using robots and intelligent systems can be applied. When robots face more complex, unstructured, and dynamic environments while expanding their workspace, the problem of Simultaneous Localization and Mapping (SLAM) becomes even more relevant and significantly harder at the same time. There is a clear and present need for efficient and most of all robust SLAM methods that are able to generate a useful map, even under erroneous data association decisions on any level in the SLAM process.

Graph-based SLAM has been the method of choice in the latest literature on SLAM in dynamic environments [1], portable SLAM systems for humans [2], SLAM with micro aerial vehicles (MAV) [3], [4], as well as underwater SLAM [5]. All of these use cases can benefit significantly from an improved robustness of graph optimization methods for SLAM. For those reasons, robust graph optimization or inference for graph-based SLAM has become a strong research focus very recently [6], [7], [8], [9], [10], [11]. These methods fall into roughly two categories:

In one ([10], [11], [8], [9] and partially [7]), incongruent graph constraints are simply discounted during optimization. This category is roughly comparable to iteratively reweighted least squares [12] or least trimmed squares [13], both traditional robust regression techniques.

The other ([6] and partially [7]) allows multiple components per constraint, either as a multimodal Mixture of Gaussians (MoG) [6], or a so-called multimodal *Max-Mixture* [7].

This paper presents a general framework for SLAM under ambiguous data associations, called Generalized Graph SLAM. Its description contains the first formal introduction of how uncertain loop closures can be modeled using hyperedges. This is extension is combined with the authors' previous work on multimodal constraints [6].

The hyperedge components allow to cope with global ambiguities, i.e. to represent multiple alternative hypotheses about possible loop closures. The multimodal components in contrast deal with local ambiguities, i.e. they handle different alternative hypotheses about the motion a robot/sensor may have undergone between two subsequent poses. This new framework of Generalized Graph SLAM can hence handle local as well as global ambiguities in a single coherent manner. Furthermore, it is shown that other robust methods found in the literature are conceptually special cases within our Generalized Graph SLAM framework.

To solve the challenges put forth in the Generalized Graph SLAM framework, a novel extension of the authors' *Prefilter* method [6] to generate good initial conditions for selecting globally consistent constraints from such multimodal hypergraphs is presented. Experiments show that the extended *Prefilter* method is both significantly more robust and faster than other robust state-of-the-art methods.

II. Related Work

The robust SLAM methods in [6], [7], [8], [9], [10], [11] improve upon traditional Graph SLAM methods by providing ways to filter out or discount incongruent graph constraints during optimization.

While not specified exactly in the paper, the $g2o$ graph optimization library by Kümmerle *et al.* [14] chooses a traditional robust optimization approach by applying an iteratively reweighted least squares (IRLS) method [12]. Their approach allows weighting the individual terms of the cost function by the computed residuals and reducing the influence of large residuals. Multiple of these robust kernels are implemented, e.g. the Huber or the Cauchy kernel [12]. Thus, incongruent constraints are weighted less, which allows the method to converge to a reasonable result when outliers are present. The latest $g2o$ version[1] was used in the experiments below.

Sünderhauf and Protzel [10], [11] chooses a more explicit reweighting scheme were the weight of a cost function term is controlled as part of the state vector during optimization. Instead of using the value of the local residual to scale the impact of it in the total cost function, switching variables are introduced that are explicitly part of the state. Either a sigmoid function [10] or a linear function [11] is used to

The authors are with the School of Electrical Engineering and Computer Science, Jacobs University, Bremen, Germany. <m.pfingsthorn, a.birk>@jacobs-university.de

[1] from https://github.com/RainerKuemmerle/g2o

weigh individual cost function terms. For the experiments below, an implementation of this method in the $g2o$ library published as open source by Sünderhauf and Protzel is used[2]. This implementation uses the more recently proposed linear switch function [11].

In the authors' previous work [6], a novel multimodal extension of the traditionally unimodal constraints used in graph-based SLAM was introduced. Standard methods assume that a constraint in the graph is inherently correct and just uncertain due to noise; it can hence be represented by a single underlying normal distribution. [6] uses a multimodal Mixture of Gaussians (MoG) instead that allows multiple mutually exclusive options (modes) in each edge. The authors also introduced the concept of local ambiguity vs. global ambiguity, where the presented approach using MoGs is used to solve the local ambiguity problem, i.e. the handling of different motion alternatives between two subsequent robot/sensor poses. Several methods are outlined in [6] to solve SLAM with locally ambiguous registration results that are expressed in multimodal MoGs, including a global graph initialization method called *Prefilter* which is shown to be very robust. It is used to discard incongruent components in the MoG constraints before optimization, which is related to least trimmed squares [13].

Olson and Agrawal [7] take a similar approach by allowing multiple single normal distributions in one constraint in the graph. However, in their method, the decision which of the constraint distributions should be used is reevaluated greedily in each step of the iteration. Instead of using a weighted sum of normal distributions as is the case with MoGs, their approach only uses the component which contributes the maximum probability to the current estimate, thus the name *Max-Mixture*. *Max-Mixtures* have the disadvantage that they are chosen greedily given the current estimate, requiring good initial conditions to allow convergence. Olson and Agrawal [7] also describe the idea that multiple loop closing constraints may be combined into one using a multimodal *Max-Mixture*, effectively representing a hyperedge but not explicitly using the name nor describing the idea in any formal way. Again, the implementation for the $g2o$ library made available by the authors Olson and Agrawal is used in the experiments below[3].

Latif *et al.* [8], [9] present a method called *RRR* to generate clusters of temporally close loop-closing constraints and check these for spatial consistency. The method makes the rather significant assumption that sequential constraints are generated using odometry and are always without outliers. A traditional $\mathcal{X}^2$ error metric is used to identify outliers in each cluster of loop-closing constraints. By explicitly making a binary decision about the inclusion or rejection of individual constraints, the method is highly related to the least trimmed squares method. By using a spatial consistency measure to select outliers, the method is also related to the *Prefilter* method above. For this method, the open source implementation for the $g2o$ library was also provided by its authors Latif *et al.*[4]

The idea of using hyperedges to represent global ambiguity (i.e. ambiguous loop detection results) is introduced in this paper, and this idea is combined with the used of multimodal edges to represent local ambiguities in the Generalized Graph SLAM framework. An extension of the multimodal *Prefilter* method that can simultaneously process multimodal MoG constraints as well as ambiguous loop closure constraints in the form of hyperedges is subsequently presented. The following section validates the presented method in comparison to the methods described above with experimental results using synthetic datasets with varying degrees of complexity. The last section concludes the paper.

III. Generalized Graph SLAM

A. Traditional Graph-Based SLAM

Formally, a pose graph is an undirected graph $G = (V, E)$ consisting of vertices V and edges E. The vertices $v_i \in V$ denote poses where the robot obtained sensor observations z_i. A pose estimate x_i is also associated with the vertex and thus is a tuple $v_i = (x_i, z_i)$. In addition to the vertices it connects, each edge $e_k \in E$ contains a constraint c_k on the pose estimates of the associated vertices, thus $e_k = (v_i, v_j, c_k)$. While the graph itself is undirected, the edge has to declare a sort of *observation direction*, the direction in which the constraint was generated, often also called the reference frame of the constraint. In case the edge is traversed in reverse *observation direction*, the constraint c must be inverted. What exactly that entails is up to the representation of the constraint.

From the formal description in [6], the joint probability of the pose graph G is

$$p(x_{1:t}|G) = \prod_{(v_i,v_j,c_k)\in E} p(x_j \ominus x_i|c_k) \tag{1}$$

where $p(x_j \ominus x_i|c_k)$ is the specific probability distribution of the constraint c_k on edge $e_k \in E$.

Usually, this is a normal distribution, so

$$p(dx|c_k) = \frac{1}{|2\pi\Sigma_k|^{1/2}} e^{-\frac{1}{2}(dx\ominus\mu_k)^T\Sigma_k^{-1}(dx\ominus\mu_k)} \tag{2}$$

This results in a very convenient negative log probability formulation that can be directly fed into a general non-linear least squares solver.

$$-\ln p(x_{1:t}|G) \approx \sum_{(v_i,v_j,c_k)\in E} (t_i^j \ominus \mu_k)^T\Sigma_k^{-1}(t_i^j \ominus \mu_k) \tag{3}$$

where $t_i^j = x_j \ominus x_i$.

B. Local Ambiguity vs. Global Ambiguity

The major source of errors in graph-based SLAM is faulty data association. Specifically, two types of data association errors are identified: a) Errors in identifying common data in two consecutive sensor observations (local ambiguity) and b)

[2]from `http://openslam.org`

[3]from `https://github.com/agpratik/max-mixture`

[4]from `https://github.com/ylatif/rrr`

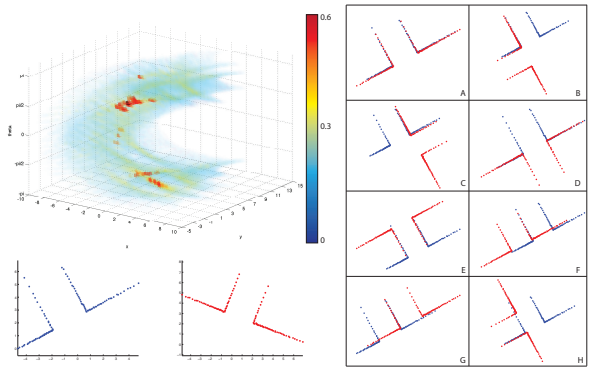


Fig. 1. A visualization of registration ambiguity, from [6]. Two simulated laser scans from the crossing of two corridors are matched with an exhaustive cross-correlation based method. Note the multiple possible registration results with high cross-correlation values on the right.

errors identifying common data in temporally distant sensor observations (global ambiguity). This section describes each of these error sources in detail, laying the foundation for the Generalized Graph SLAM framework.

Specifically, the term ambiguity is used for the situation where a clear global optimum of the respective registration cost function in the local case or the data association metric in the global case can not be found, but multiple local optima are present instead.

Local ambiguity corresponds to the case where two consecutive sensor observations exhibit multiple possible registration results, such as in the example shown in figure 1. A registration result of such two scans is called locally ambiguous if these ambiguities can not be resolved using only information present in the observations itself. In other words, the registration cost function has multiple optima, resulting in a multimodal Mixture of Gaussian (MoG) probability distribution for the corresponding constraint.

$$p(x_i|x_{i-1}) = \sum_{m=1}^{M_k} \pi_m \mathcal{N}(x_i \ominus x_{i-1}|\mu_m, \Sigma_m) \qquad (4)$$

with $\sum \pi_m = 1$. Each mean μ_m corresponds to an optimum in the registration cost function, Σ_m corresponds to the inverse of the hessian at that point, and the weight π_m should be proportional to the value of the registration cost function at μ_m.

Global ambiguity corresponds to the case of uncertain loop closures, where two temporally distant sensor observations may or may not show the same section of the environment. Formally, there exists a probability mass function (PMF) which is defined over all previously inserted vertices in the graph, and a null hypothesis in case the current vertex is completely new. This PMF can be represented as the weights π_m of a more generalized mixture over all previous poses and an uninformative uniform distribution representing the null hypothesis.

$$p(x_i|x_{1:i-1}) = \pi_0 \mathcal{U}(\mathbb{R}^d) + \sum_{j=1}^{i-1} \pi_j p(x_j \ominus x_i|c_j) \qquad (5)$$

with $\sum_0^{i-1} \pi_j = 1$, and where $x_{1:i-1}$ are all poses from vertex v_1 to v_{i-1}, $p(dx|c)$ is any probability distribution representing a registration result, π_0 is the weight of the null hypothesis, and $\mathcal{U}(\mathbb{R}^d)$ is the uniform distribution over all real numbers of the same degree of freedom d as the poses.

Note that local ambiguity may occur also in the registration result referenced in the global ambiguity case. Thus they describe orthogonal problems, both or either may or may not occur in any given SLAM problem.

This means that solutions to both are required, though in the past both have been neglected in favor of a simple traditional unimodal SLAM model. However, many limitations of traditional unimodal SLAM methods that do not use explicit modeling of both sources of faulty data association have been noted, most of which had been attempted to solve by more complex and involved SLAM front ends. These front ends would filter out outliers in both the local and global case and only present verified registration results (local disambiguation) and loop constraints (global disambiguation) to the SLAM backend for optimization. In situations where either local or global ambiguity occurred, these methods would reject all results, potentially eliminating useful information.

C. Modeling Uncertain Loop Constraints as Hyperedges

Modeling the underlying probability mass function from equation 5 exhaustively for all previous poses is wasteful. Most of the weights π_j will be zero or very close to zero. Instead, a more compact representation is needed to exploit this *sparse* connectedness to older vertices.

Graph theory presents a fitting concept in this case, namely a hyperedge. Formally, a hyperedge is a set of vertices that are connected. Thus, instead of every edge $e \in E$ consisting of exactly two endpoints v_i, v_j and the associated constraint c as described in section III-A, a hyperedge in a pose graph is defined as a tuple $e = (v_i, N, \{v_j\}, \{\pi_j\}, \{c_j\})$. The weight of the null hypothesis is implicitly given by $\pi_0 = 1 - \sum \pi_j$, with $\sum \pi_j \leq 1$.

Note that due to the geometric interpretation of the edge, an *observation direction* is still necessary, so v_i is defined as the *reference* of the hyperedge and the base frame of the relative poses represented in the constraints. $\{v_j\}$ is the set of vertices the reference vertex v_i is connected to by this edge, and $N = |\{v_j\}|$. Note that the case where $N = 1$, i.e. there is no ambiguity which older vertex v_j the reference vertex v_i should be connected to, is explicitly covered as well, while still allowing for discounting of this one constraint by reducing the weight π_1.

For the general case, equation 5 becomes

$$p(x_i|G) = \prod_{e \in E} \left[(1 - \sum_{j=1}^{N} \pi_j) \mathcal{U}(\mathbb{R}^d) + \sum_{j=1}^{N} \pi_j p(x_j \ominus x_i|c_j) \right] \qquad (6)$$

$$\approx \prod_{e \in E} \sum_{j=1}^{N} \pi_j p(x_j \ominus x_i|c_j) \qquad (7)$$

Since $\mathcal{U}(\mathbb{R}^d)$ is practically zero everywhere (technically $\frac{1}{\infty}$), the term corresponding to the null hypothesis is dropped. This means that the expression looks exactly like a regular mixture, with the difference that $\sum_{j=1}^{N} \pi_j \leq 1$ instead of $\sum_{j=1}^{N} \pi_j = 1$.

In the following, each $p(x_j \ominus x_i | c_j)$ is called a *hypercomponent* to distinguish between components in hyperedge and MoG constraint mixtures. π_j will be referred to as the hypercomponent weight.

Without loss of generality, the Generalized Graph SLAM framework assumes that all edges in a generalized pose graph are hyperedges and all the constraints c_j of each edge are multimodal MoG constraints. Here, the cases with $N = 1$ (i.e. no global ambiguity) and $M_k = 1$ (i.e. no local ambiguity) are explicitly included. For this generalized graph, the joint probability then becomes (extended from eq. 14 in [6])

$$\begin{aligned} p(x_{1:t}|G) &= \prod_{e \in E} \sum_{j=1}^{N} \pi_j \sum_{m=1}^{M_k} \pi_m p(t_i^j | \mu_m, \Sigma_m) \\ &\text{with } e = (v_i, N, \{v_j\}, \{\pi_j\}, \{c_j\}) \\ &\text{and } t_i^j = x_j \ominus x_i \end{aligned} \tag{8}$$

Similarly for the joint log probability

$$\ln p(x_{1:t}|G) = \sum_{e \in E} \ln \left[\sum_{j=1}^{N} \pi_j \sum_{m=1}^{M_k} \pi_m p(t_i^j | \mu_m, \Sigma_m) \right] \tag{9}$$

An equivalent formulation moves the hypercomponent weights into the MoG sum, allowing the same vertex multiple times in the set $\{v_j\}$:

$$\begin{aligned} p(x_{1:t}|G) &= \prod_{e \in E} \sum_{l=1}^{L} \pi_l p(t_i^j | \mu_l, \Sigma_l) \\ &\text{with } e = (v_i, L, \{v_j\}, \{\pi_l\}, \{(\mu_l, \Sigma_l)\}) \end{aligned} \tag{10}$$

where $L = \sum M_k$ and $\pi_l = \pi_j \pi_m$ for the l-th hypercomponent/MoG component combination. Again, $\pi_0 = 1 - \sum \pi_l$. This formulation, though with $\sum \pi_l = 1$, is implicitly used in Olson's *Max-Mixture* method [7], though not explicitly described. However, eq. 8 is conceptually clearer as it presents a clear separation of global and local ambiguity. Furthermore, there is of course the main challenge not only to represent local and global ambiguities but to find a robust and efficient optimization method for them, which may require separate models for each.

There are now some considerations to be made for computing the natural logarithm of the double sum of weighted Gaussians. In the special case of a simple unimodal edge, where $N = 1$, $\pi_1 = 1$, and $M_k = 1$,

$$\begin{aligned} &\ln \left[\sum_{j=1}^{N} \pi_j \sum_{m=1}^{M_k} \pi_m p(t_i^j | \mu_m, \Sigma_m) \right] \\ &= \ln p(t_i^j | \mu_m, \Sigma_m) \\ &= -\frac{1}{2} \ln \left(|2\pi\Sigma_1| \right) - \frac{1}{2} (t_i^j \ominus \mu_1)^T \Sigma_1^{-1} (t_i^j \ominus \mu_1) \end{aligned}$$

In the following, such an edge will be referred to as *simple*.

In the special case of a purely multimodal edge, where $N = 1$, $\pi_1 = 1$,

$$\begin{aligned} &\ln \left[\sum_{j=1}^{N} \pi_j \sum_{m=1}^{M_k} \pi_m p(t_i^j | \mu_m, \Sigma_m) \right] \\ &= \ln \left[\sum_{m=1}^{M_k} \pi_m p(t_i^j | \mu_m, \Sigma_m) \right] \end{aligned}$$

In the previous two cases, a $\pi_1 < 1$ will result in a simple addition by $\ln \pi_1$ in the log-probability.

In the special case of a pure hyperedge with unimodal subcomponents, where $M_k = 1$,

$$\begin{aligned} &\ln \left[\sum_{j=1}^{N} \pi_j \sum_{m=1}^{M_k} \pi_m p(t_i^j | \mu_m, \Sigma_m) \right] \\ &= \ln \left[\sum_{j=1}^{N} \pi_j p(t_i^j | \mu_m, \Sigma_m) \right] \end{aligned}$$

Additionally, there are a number of methods described in recent literature that can be treated as special cases of the Generalized Graph SLAM framework. The special case where $N = 1$ and c_1 is a unimodal Gaussian constraint (i.e. $M_k = 1$) corresponds to the work done by Sünderhauf and Protzel [10]. In this case, $\pi_1 = \omega_{ij} = sig(s_{ij})$, where s_{ij} is the switch variable between vertices v_i and v_j (see eq. 7 in [10]), or $\pi_1 = \Psi(s_{ij}) = s_{ij}$ for the linear case (cf. eq. 1 in [11]). Similarly, the *RRR* algorithm of Latif *et al.* [8], [9] makes a strictly binary decision where Sünderhauf and Protzel make a fuzzy one, and thus can be modeled the same way in this framework, i.e. $\pi_1 \in 0, 1$.

The special case where the weights π_j and π_m are adjusted at every iteration such that only one π_j and π_m retain their original value, i.e.

$$(j^*, m^*) = \underset{j,m}{\operatorname{argmax}} \, \pi_j \pi_m p(dx | \mu_m, \Sigma_m) \tag{11}$$

$$\pi_j = \begin{cases} \pi_j & \text{if } j = j^* \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

$$\pi_m = \begin{cases} \pi_m & \text{if } m = m^* \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

corresponds to the *Max-Mixture* method (see eq. 4 in [7]). Olson and Agrawal aggregate the two conceptually separate weights π_j and π_m into one in their discussion, as in the equivalent formulation presented in eq. 10. As such, they do not offer an implicit null hypothesis choice, but the mixture has to explicitly include a normal distribution defined to be the null hypothesis, which usually has a very large covariance.

The *Prefilter* method [6], and its extension to this Generalized Graph SLAM framework described in the next section, is used to make a similar selection of weights as *Max-Mixture* does. Weights are set such that exactly one π_j and one π_m per edge is one, indicating the component that explains the

estimate generated by the *Prefilter* method best:

$$(j^*, m^*) = \underset{j,m}{\operatorname{argmax}}\, \pi_j \pi_m p(dx|\mu_m, \Sigma_m) \tag{14}$$

$$\pi_j = \begin{cases} 1 & \text{if } j = j^* \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

$$\pi_m = \begin{cases} 1 & \text{if } m = m^* \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

However it is done once before the optimization starts and this decision is not changed during optimization, allowing the use of standard optimization methods.

D. Good Initial Conditions on Multimodal Hypergraphs

Algorithm 1: $ExpandMultimodal(\mathbf{T}, t, v, v_{next}, c)$

Input: List of traversal states $\mathbf{T}$, current traversal state t, current vertex v, next vertex v_{next}, multimodal constraint c
Output: Modified list of traversal states $\mathbf{T}$

for *every multimodal component* m *in* c **do**
  make a new traversal state t_{new} as a copy of t;
  x = pose of v in $t.\mathbf{X}$;
  $t_{new}.\mathbf{X} = t_{new}.\mathbf{X} \bigcup (x \oplus \mu_m)$
  for *every edge* e_{adj} *adjacent to* v_{next} *that is not in* $t_{new}.E_{used}$ **do**
    enqueue($t_{new}.P$, (v_{next},e_{adj}));
    $t_{new}.E_{used} = t_{new}.E_{used} \bigcup \{e_{adj}\}$;
  end
  append t_{new} to $\mathbf{T}$;
end

Algorithm 2: $edgeweight(e)$

Input: MoG Hyper PoseGraph edge $e \in E$
Output: computed edge weight ω

$\omega = 0$;
for *all constraints* c_j *in* e **do**
  $\omega = \omega + M_k$;
end
return ω;

The extension of the original *Prefilter* algorithm [6] to also handle hyperedges in addition to multimodal ones is rather straight forward.

Algorithm 3 shows the pseudocode for the *Prefilter* method extended to hypergraphs. The main difference to the original *Prefilter* is that through choosing the hypercomponent to follow, the underlying graph topology for each sample changes. Therefore, the state of the whole minimum spanning tree traversal has to be kept associated with the corresponding pose sample in a list of traversal state $\mathbf{T}$. For simplicity, each MoG component also gives rise to a new traversal state instance, even though they do not change the graph topology and some data is duplicated. This simplification also allows straightforward parallelization of

Algorithm 3: The *Prefilter* algorithm.

Input: MoG Hyper PoseGraph G, maximum number of hypotheses N
Output: $\mathbf{X}$: a set of N sets of vertex poses $X = \{x_i\}$

1 initialize an empty list $\mathbf{T}$ of traversal states;
2 let t be a traversal state;
3 $t.\mathbf{X} = \{x_1\}$;
4 $t.V_{used} = \{v_1\}$;
5 $t.E_{used} = \emptyset$;
6 initialize priority queue $t.P$ to sort by $edgeweight(e)$;
7 **for** *all adjacent edges* e *of* v_1 **do**
8 enqueue($t.P$, (v_1, e));
9 $t.E_{used} = t.E_{used} \bigcup \{e\}$;
10 **end**
11 append t to $\mathbf{T}$;
12 **while** $\exists t \in \mathbf{T} : t.P$ *not empty* **do**
13 **for** $\forall t \in \mathbf{T} : t.P$ *not empty* **do**
14 (v, e) = dequeue($t.P$);
15 **if** $v = e.v_{start}$ **then**
16 **for** *every hyperedge component* j **do**
17 $ExpandMultimodal(\mathbf{T}, t, v, v_j, c_j)$;
18 **end**
19 **else**
20 let j be the hyperedge component of e where $v_j = v$;
21 $ExpandMultimodal(\mathbf{T}, t, v_j, v, invert(c_j))$;
22 **end**
23 **if** $\sum_{j=1}^{N} e.\pi_j = 1$ **then**
24 remove t from $\mathbf{T}$;
25 **end**
26 **end**
27 **if** $|\mathbf{T}| > N$ **then**
28 sort $\mathbf{T}$ by joint probability of assigned vertex poses $\mathbf{X}$ of each element;
29 truncate $\mathbf{T}$ to contain only the N most probable elements;
30 **end**
31 **end**
32 $\mathbf{X} = \bigcup_{t \in \mathbf{T}} t.\mathbf{X}$;

the algorithm for large values of N and complex graphs. However, the implementation used in the experiments is single threaded to allow a fair comparison.

Note that the null hypothesis is never directly referenced in the algorithm. By design of the algorithm, keeping an unmodified version of the current traversal state t in the list $\mathbf{T}$ corresponds to the case where the current edge e is not used, i.e. where the null hypothesis is chosen. This works since edges are marked as used when they are enqueued in the priority queues, and dequeueing an edge from t without using it effectively deletes it from the graph topology for t. Furthermore, calling $ExpandMultimodal(\dots)$ does not change the passed current traversal state, only new traversal states are generated corresponding to all modes. This means that by line 23 in algorithm 3, the current traversal state t is

unchanged, and thus corresponding to the null hypothesis. Line 23 checks if the null hypothesis is inadmissible by checking its weight, and if it is not, removes t from $\mathbf{T}$, thus not following the null hypothesis.

The final set of sorted vertex poses $\mathbf{X}$ can be used to select not only components from a MoG, as described in [6], but also hyperedge components in the same way.

IV. Experiments and Results

#	$C(G)$	#edges with X modes/hypercomponents X=1	X=2	X=3	X=4	X=5	%
1	1	255	0/1	0	0	0	0.4
2	2	254	1/1	0	0	0	0.8
3	3	253	2/1	0	0	0	1.2
4	4	252	2/2	0	0	0	1.6
5	8	248	4/4	0	0	0	3.2
6	16	240	8/8	0	0	0	6.4
7	32	224	16/16	0	0	0	12.8
8	64	192	32/32	0	0	0	25
9	82.72	192	16/16	16/16	0	0	25
10	105.36	192	8/8	8/8	16/16	0	25
11	126.99	192	4/4	4/4	8/8	16/16	25

TABLE I

The 11 complexity classes used in the experiments.

A synthetic data set was generated, much like the one used in [6]. Each generated graph consists of 128 vertices and 256 edges. The aim of the experiments is to find out how robust the methods are with respect to ambiguities in the data, so a number of pose graphs with different complexities were generated. Specifically, the complexity metric introduced in [6] was extended to encompass hyperedges as well:

$$C(G) = \log_2 \left(\prod_{e \in E} \sum_{j=1}^{N} M_j \right) = \sum_{e \in E} \log_2 \left(\sum_{j=1}^{N} M_j \right) \quad (17)$$

This way, a hyperedge with n unimodal hypercomponents has the same complexity as a hyperedge with just one hypercomponent containing a MoG constraint with n modes. The metric hence captures the fact that hypercomponents and MoG components represent different alternative spatial relations between nodes in the graph that can lead to a combinatorial explosion. A detailed analysis over different graph topologies and complexities would not have been possible with existing benchmark datasets.

Table I shows a summary of generated complexity classes and the distribution and number of components in the MoG constraints and hyperedges. The overall percentage of non-simple edges is also shown. A total of 110 graphs were generated in 11 classes with an increasing complexity, 10 per class. The first 7 classes only contain MoG constraints or hyperedges with two components in varying numbers. In class 3, for example, the graphs contain two MoG constraints and one hyperedge, both with two components ($X = 2$ in the table). Classes 9 through 11 do not add more non-simple edges, but more MoG or hyperedge components.

Instead of generating a completely new random graph for the more complex classes, the already generated less complex

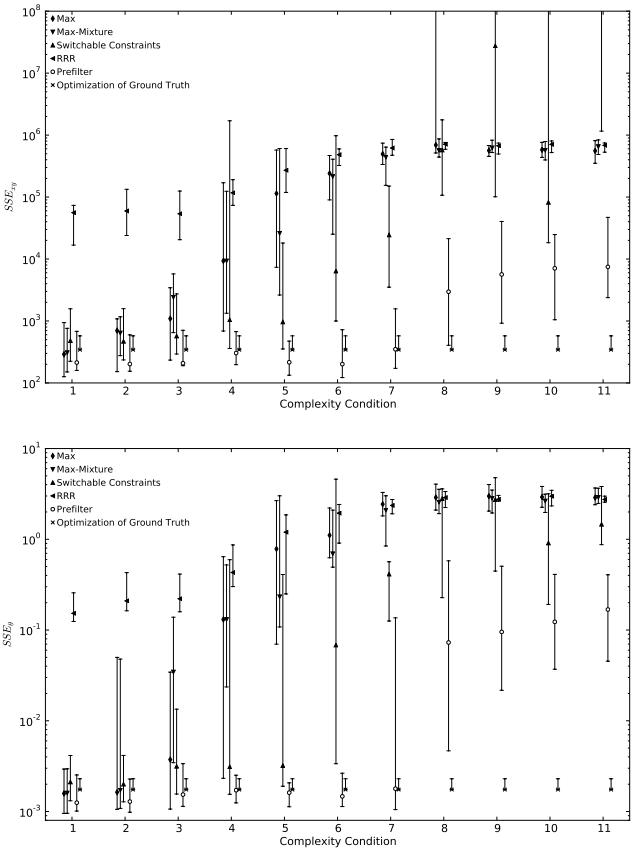


Fig. 3. Final SSE metric relative to ground truth for each of the 11 complexity classes. The median and upper/lower quartiles are shown. Note the log scale on the y axis. The final SSE metric of the optimization result using the ground truth graph is also shown for comparison.

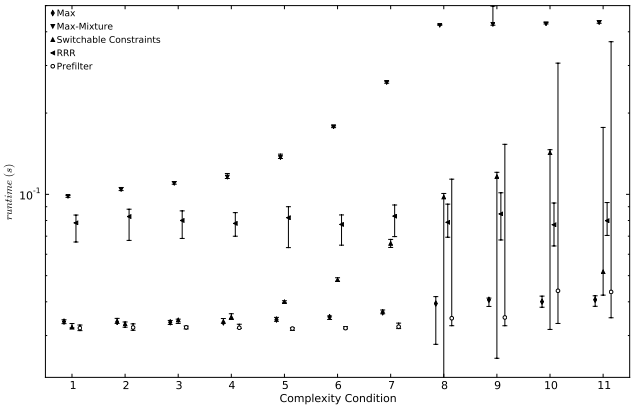
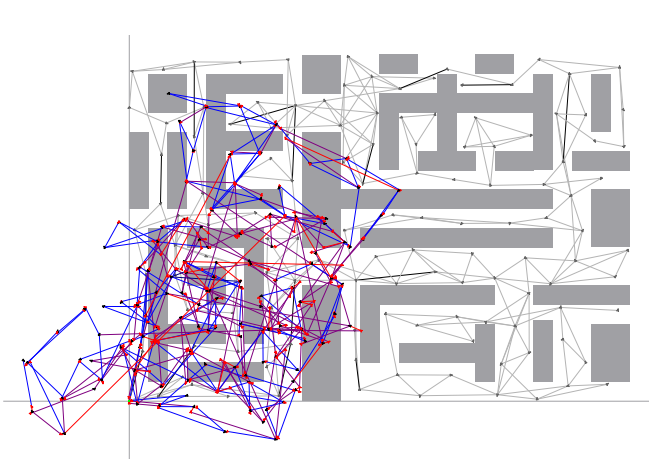
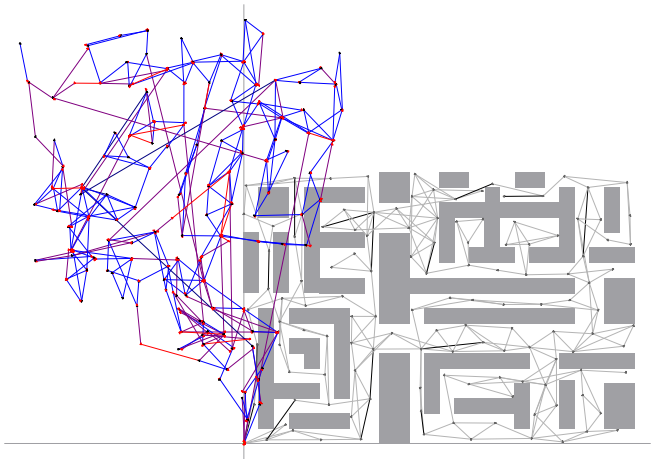


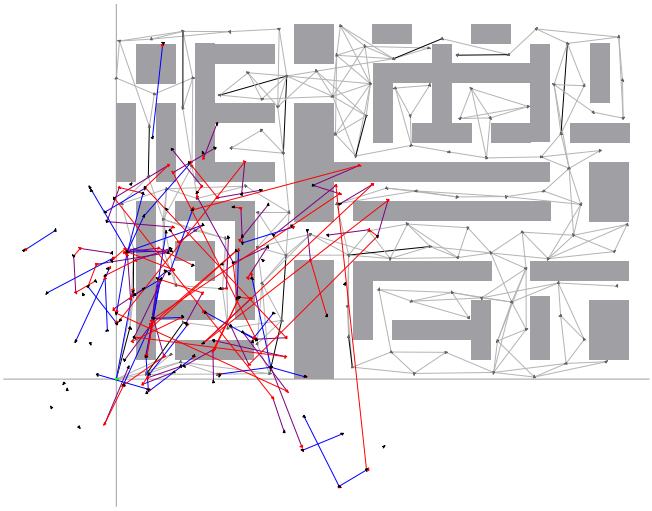
Fig. 4. Runtimes for each of the methods on all 11 complexity classes. The median and upper/lower quartiles are shown. Times were recorded on an Intel i7-3770 3.4GHz with 16GB RAM. Note the log scale on the y axis. Runtimes varied from $0.01s$ to $1.45s$ over all methods, quartiles are between $0.03s$ and $0.44s$.



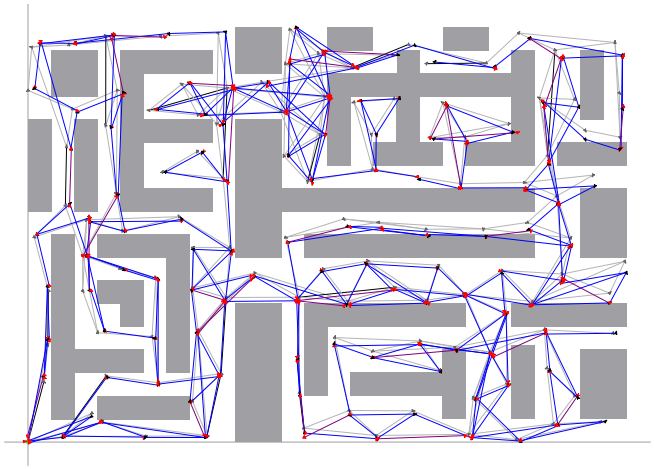
(a) Optimization result with *Max-Mixture* from the traditional breadth-first initialization, $SSE_{xy} = 631554$, $SSE_{\theta} = 4.75323$.



(b) Optimization result with *Switchable Constraints* also from breadth-first initialization, $SSE_{xy} = 1162181$, $SSE_{\theta} = 2.02238$.



(c) Optimization result with *RRR* also from breadth-first initialization, $SSE_{xy} = 625717$, $SSE_{\theta} = 4.31798$.



(d) Robust Gauss-Newton optimization result using the MoG and hyperedge components chosen by *Prefilter*, $SSE_{xy} = 168.343$, $SSE_{\theta} = 0.00173$.

Fig. 2. One example graph of the data set of complexity class 7 and $C(G) = 32$, with a total of 16 multimodal MoG edges with two components and 16 hyperedges with two hypercomponents. Ground truth is shown in gray in the background.

graphs were reused. Thus, a total of 10 base graphs were generated consisting completely of simple edges. For class 1, one hypercomponent was added to a random simple edge of the base graph. For class 2, one MoG component was added to a random simple edge of the same graph. For class 3, one MoG component was added to another random simple edge, and so on. For more complex classes, existing components on either hyperedges or MoG constraints were reused and additional components added where necessary. This way, the differences between the graphs in increasing complexity are minimal, and thus the performance difference of the methods is solely due to the additional components in either MoG constraints or hyperedges.

Five main methods were tested and compared. The state-of-the-art *Max-Mixture* [7], *Switchable Constraints* [10], [11], and *RRR* [9] methods were used as a comparison basis to evaluate the extended *Prefilter* method described above. This is mostly a comparison of initialization methods, as *Max-Mixture* is very sensitive to the initial condition, as also noted by the authors of [7]. The *Switchable Constraints* method is less susceptible, but still suffers significantly from bad initial conditions. *RRR* is supposed to be independent of the initial configuration, but fails to even maintain the connectedness of the graph. A fifth method called *Max* representing the traditional case where only the most weighted component (hypercomponent or MoG component) is chosen, i.e. $j^*, m^* = \operatorname{argmax} \pi_j \pi_m$, is also used as a baseline for the other methods [6]. This approach models the case where an unreliable loop detection method and unimodal registration method is used. A traditional breadth-first initialization was performed before optimization with either of these methods.

The result of *Prefilter* was used to select components of all hyperedges and MoG constraints analogous to the way described in [6] for optimization with a standard robust optimization method implemented in the $g2o$ library [14].

The same solver, a Gauss-Newton method implemented in

$g2o$, was used for all three approaches, so their convergence and computational complexity can be fairly evaluated. For the *Max*, *Max-Mixture*, and *Prefilter* methods, a *Cauchy* robust kernel with a width of 1 was used. *Switchable Constraints* implements explicit reweighting, so an implicit one with a robust kernel was not used. The *RRR* implementation hardcoded its solver of choice, which was left as is. All methods other than *RRR* were run for 150 iterations. *RRR* does not have an explicit parameter to control the number of iterations. The odometry rate and loop closure rate parameters of *RRR* were both set to 1.

Figure 3 shows the median and upper and lower quartiles of the final SSE_{xy} and SSE_{θ} error [15] relative to Ground Truth of the ten sample pose graphs per complexity class after optimization using all investigated methods. Note the log scale of the y axis. As a comparison of achievable results with Ground Truth initialization and no outliers, the final SSE errors of the optimized Ground Truth base graphs is also included. This optimization did not use a robust kernel, therefore it is surprising, but not impossible that the Ground Truth optimization result has a higher final error than some of the other methods. It is obvious that, even though there is a high variance for all methods, *Prefilter* performs multiple orders of magnitude better than *Max-Mixture*, and around an order of magnitude better than *Switchable Constraints*. This especially holds in highly complex classes. *Switchable Constraints* exhibits a slightly better robustness towards graph complexity than *Max-Mixture*, though it also exhibits a very large variance in the results. Surprisingly, *RRR* fails at all graphs, and changing any of the exposed parameters (odometry and loop rate) has no effect. This happens because a very significant number of constraints are falsely rejected, which breaks the graph (see figure 2c).

Figure 4 shows the runtimes of the compared methods. Again, note the log scale of the y axis. The required runtime of the *Max-Mixture* method increases significantly with the graph complexity. A similar trend is evident in the time required by the *Switchable Constraints* method, note the large median runtime. The *Prefilter* method only needs more computational time occasionally with very complex graphs, indicated by the low median required time for this method even at high complexities. However, at these complexities (classes 8-11), the other methods no longer converge to satisfactory results at all. Thus longer runtimes of *Prefilter* relative to the less complex classes are definitely worth the significant gain in robustness over the other methods.

V. Conclusions

In this paper, the Generalized Graph SLAM framework was presented. Especially, a formal description of how to use hyperedges to encode uncertain loop closures was introduced. This method to handle global ambiguities is combined with multimodal edge constraints to cope with local ambiguities. Current state-of-the-art methods in robust graph-based SLAM were shown to be special cases of this Generalized Graph SLAM framework.

A method to generate good initial conditions for the most general case of multimodal hypergraphs was presented and validated with experiments on synthetic graphs. The experiments showed that this extended *Prefilter* method is both significantly more robust and less computationally demanding than current state-of-the-art approaches.

Acknowledgements

The research leading to the presented results was supported in part by the European Community's Seventh Framework Programme under grant agreement n. 270350 "Cognitive Robot for Automation Logistics Processes (RobLog)", and grant agreement n. 288704 "Marine robotic system of self-organizing, logically linked physical nodes (MORPH)".

References

[1] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. Leonard, "Dynamic pose graph slam: Long-term mapping in low dynamic environments," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, oct. 2012, pp. 1871 –1878.

[2] M. Fallon, H. Johannsson, J. Brookshire, S. Teller, and J. Leonard, "Sensor fusion for flexible human-portable building-scale mapping," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, oct. 2012, pp. 4405 –4412.

[3] F. Fraundorfer, L. Heng, D. Honegger, G. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor mav," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, oct. 2012, pp. 4557 –4564.

[4] R. Leishman, J. Macdonald, T. McLain, and R. Beard, "Relative navigation and control of a hexacopter," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, may 2012, pp. 4937 –4942.

[5] M. Pfingsthorn, A. Birk, and H. Bülow, "Uncertainty estimation for a 6-dof spectral registration method as basis for sonar-based underwater 3d slam," in *Robotics and Automation, 2012. Proceedings. ICRA '12. IEEE International Conference on*. IEEE Press, 2012.

[6] M. Pfingsthorn and A. Birk, "Simultaneous Localization and Mapping with Multimodal Probability Distributions," *The International Journal of Robotics Research*, vol. 32, no. 2, pp. 143–171, 2013. [Online]. Available: http://ijr.sagepub.com/content/32/2/143.abstract

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

[8] Y. Latif, C. Cadena, and J. Neira, "Realizing, reversing, recovering: Incremental robust loop closing over time using the irrr algorithm," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct., pp. 4211–4217.

[9] Y. Latif, C. C. Lerma, and J. Neira, "Robust loop closing over time," in *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.

[10] N. Sunderhauf and P. Protzel, "Towards a robust back-end for pose graph slam," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, may 2012, pp. 1254 –1261.

[11] ——, "Switchable constraints for robust pose graph slam," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, oct. 2012, pp. 1879 –1884.

[12] P. J. Huber and E. M. Ronchetti, *Robust Statistics*, 2nd ed. John Wiley & Sons, Inc., March 2009.

[13] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*. John Wiley & Sons, Inc., 2005. [Online]. Available: http://dx.doi.org/10.1002/0471725382.fmatter

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

[15] E. Olson, "Robust and efficient robotic mapping," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, June 2008.