Open access • Journal Article • DOI:10.1177/0278364913498910

Robust loop closing over time for pose graph SLAM — Source link

Yasir Latif, Cesar Cadena, José Neira

Institutions: University of Zaragoza, George Mason University

Published on: 01 Dec 2013 - The International Journal of Robotics Research (SAGE Publications)

Topics: Graph (abstract data type), Odometry and Mobile robot

Robust Loop Closing Over time for Pose Graph SLAM

Yasir Latif, César Cadena and José Neira [*][†]

Abstract

Long term autonomous mobile robot operation requires considering place recognition decisions with great caution. A single incorrect decision that is not detected and reconsidered can corrupt the environment model that the robot is trying to build and maintain. This work describes a consensus-based approach to robust place recognition over time, that takes into account all the available information to detect and remove past incorrect loop closures. The main novelties of our work are: (1) the ability of realizing that, in light of new evidence, an incorrect past loop closing decision has been made; the incorrect information can be removed thus recovering the correct estimation with a novel algorithm; (2) extending our proposal to incremental operation; and (3) handling multi-session, spatially related or unrelated scenarios in a unified manner. We demonstrate our proposal, the RRR algorithm, on different odometry systems, e.g. visual or laser using different front-end loop closing techniques. For our experiments we use the efficient graph optimization framework g2o as back-end. We back our claims with several experiments carried out on real data, in single and multi-session experiments showing better results than those obtained by state of the art methods, comparisons against whom are also presented.

1 Introduction

An autonomous mobile robot carrying out an indefinitely long task can be frequently moving around the same space. This allows the robot to repeatedly observe the same location at different moments in time. Recognizing revisited places, usually the task of what is called the "front-end" of these systems, allows to reduce the overall uncertainty caused by drift in odometry. It also improves the precision in the environment model that the robot may be creating and maintaining, usually the responsibility of the "back-end" of these systems. This is the well know and well studied problem of "loop closing" or "place recognition". Many algorithms have been proposed that detect such revisits. All are prone to perceptual aliasing, i.e. different places appear very similar to the sensor that is perceiving the environment. This prevents these algorithms from achieving 100% accuracy over long periods of time. This is a serious drawback because, while correct place recognition improves the model estimate, incorrectly associating two different places can severely corrupt the map estimate. An example of an incorrect loop closure due to perceptual aliasing can be seen in Fig. 1. Loop closures, as suggested by a Bag of Words approach, are shown along with ground truth from five runs from the Bicocca dataset [20]. The loop closures not parallel to the z-axis are incorrect.

In order to deal with the problem of corrupt map estimates, we need to be able to distinguish between correct and incorrect loop closures being introduced by the front-end place recognition

[*]This research has been funded by the Dirección General de Investigación of Spain under projects DPI2009-13710 and DPI2009-07130 and by DGA-FSE(group T04), and supported by US Army Research Office Grant W911NF-1110476. Preliminary versions of this work were presented in [14] and [13].

[†]Yasir Latif and José Neira are with the Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Zaragoza 50018, Spain. {ylatif, jneira}@unizar.es. César Cadena is with the Computer Science Department, Volgenau School of Engineering at George Mason University, Fairfax, VA 20030, US. ccadenal@gmu.edu.

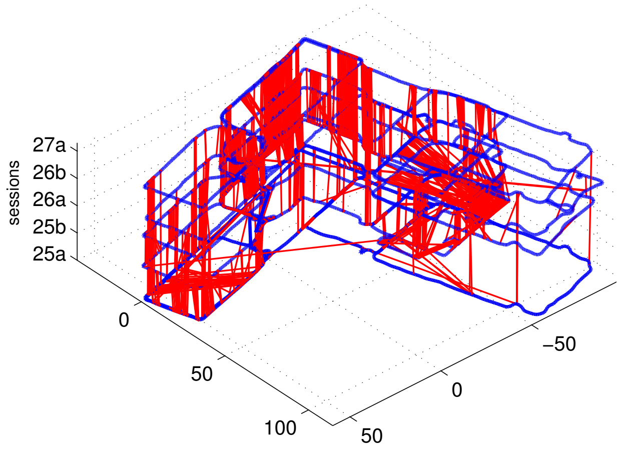


Figure 1: Loop Closing from a Bag of Words loop closure detection method. The ground truth trajectory for five runs (blue) and loop closing suggested by the place recognition system (red). Loop closings not parallel to the z-axis are incorrect.

algorithm. Once such a distinction has been made, incorrect loop closures can be discarded from the estimation process and a correct map estimate can be recovered. This is especially important in the case of long term mapping/autonomy because repeatedly revisiting the same place allows better detection and correction of mistakes in place recognition. Redundancy allows increased reliability in the verification of loop closure constraints.

We propose a novel algorithm: *Realizing, Reversing, and Recovering* (*RRR*) for loop closure verification: given one or more sets of sequential constraints provided by odometry and a set of potential loop closing constraints provided by a place recognition system, the algorithm is able to differentiate between the correct and incorrect loop closures. The underlying idea is that of consensus: correct loop closures tend to agree among themselves and with the sequential constraints, while incorrect ones do not. This, along with convergence properties of optimization techniques, provides a robust method for rejecting false loop closures.

The *RRR* algorithm is able to realize that the place recognition system has generated wrong constraints, remove them if necessary, and recompute the state estimation. More specifically this algorithm has two main characteristics:

- it makes the best possible loop closing decisions given all the available information provided by the place recognition and the odometry systems, either incrementally or in batch; and
- it has the ability to make decisions over loop closures among spatially connected or unconnected sessions in a unified manner

Our method works with the pose graph formulation. It is a part of the back-end of a Simultaneous Localization and Mapping (SLAM) system and is therefore independent of the type of sensor used for odometry or loop closing. All our method needs is a system that is able to generate a pose graph from the sequential pose constraints, and a place recognition system for the non-consecutive loop closure constraints. The output of our method is an optimized pose graph, with no auxiliary constraints and without changing the formulation of the problem.

In the next section we discuss work relevant to this problem and highlight the need for a robust loop closing detection method. In section 3 we detail our proposal, the RRR algorithm, its incremental variant, iRRR, and its multi-session capabilities. Experiments are detailed in section 4 along with comparisons and evaluations with competing state of the art methods. Finally, in section 5 and 6 further discussion and conclusions about this work are presented along with directions of future work.

2 Related work

Several approaches to persistent mapping have appeared in the robotics literature in the recent past. Konolige and Bowman [10] presented a stereo visual odometry system that works together with a bag of words place recognition system to build multiple representations of dynamic environments over time. Multiple map stitching, recovery from odometry failures, loop closing, and global localization, all rely on the robustness of the place recognition system. "Weak links" are removed when place recognition is able to close loops, making it prone to errors when the place recognition closes loops incorrectly. Similarly, McDonald et al. [15] presented a multi-session 6 DOF Visual SLAM system using "anchor nodes". In their approach place recognition is assumed to be perfect and its output is trusted every time. Sibley et al. [22] presented a relative bundle adjustment approach for large scale topological mapping. They show an example of mapping from London to Oxford, over a trajectory of about $121km$. They also use appearance based place recognition and are therefore in danger of making wrong decisions in case of incorrect place recognition. Walcott-Bryant et al. [27] presented a mapping solution for low dynamic environments, namely DG-SLAM, in which outdated information is pruned based on new observations. Their method currently assumes a known starting and ending point for the robot in order to avoid errors in alignment of multiple sessions. All of these approaches to large scale and persistent mapping rely on a place recognition system with zero false positives.

Different works have been developed to avoid or at least minimize the effect of wrong constraints. One popular way is to robustify the front-end place recognition system. Appearance based loop closing approaches that work in real time usually follow the bag of words approach proposed by Sivic and Zisserman [23]. FabMap, from Cummins and Newman [4], uses a bag of words approach with probabilistic reasoning that has been shown to work robustly over a long trajectory. The BoW-CRF system from Cadena et al. [2] generates loop closings using a bag-of-words and carries out verifications using conditional random fields. This improves the robustness of the system. In any case, neither approach can guarantee 100% accuracy. Olson et al. [18] proposed a simple per-loop verification method that uses a depth-limited search to identify loops and rejects loop closures that encode a transformation that varies significantly from identity. Traditionally, simple per-loop closure techniques measures the "loopiness" of the loop closure being proposed. In other words, they measure how close the transformation in the loop is to Identity and validity is established based on a chi squared test. The problem with this approach is that large loops have drift in them and they tend to fall outside the threshold. Similarly, Olson [16] proposed a hypothesis verification method for loop closure constraints using graph partitioning based on spectral clustering. The measure of "loopiness" is calculated with neighbouring loop closures followed by spectral analysis to find consensus. A cluster that is topologically incorrect, but consistent within itself may end up being chosen. Also in our experiments we have found that if there is a large cluster, it dominates all other clusters and ends up being the only cluster selected.

Given that in some environments the perceptual aliasing is almost impossible to avoid, even for human beings, at the moment it seems that all these improvements in front-ends are insufficient to guarantee robustness.

Another approach is to delay decision making and maintain multiple topologies of the map with an associated belief for each one. Ranganathan and Dellaert [19] follow this approach using a

Rao-Blackwellized particle filter. However, they do not explicitly show how their system is affected by the presence of incorrect loop closures and how to recover from them. Their method is also unique in the sense that it uses the estimation process itself to reason about possible loop closures.

An alternative strategy is go to the optimizer itself, the back-end. The majority of pose graph optimizers assume constraints with Gaussian noise. For this kind of problems, different improvements have been developed in presence of non-Gaussian noise, e.g. using robust cost functions (Huber functions) [7]. In cases where the back-end is a filter-based approach, Agamennoni et al. [1] proposed a outlier-robust Kalman filter. Unfortunately, these approaches can only reduce the effect of outliers in the final estimation and they fail in presence of multiple or persistent outliers since each observation is treated on its own and there is no notion of overall consistency.

In recent literature, two methods have been proposed to deal with the problem of loop closure verification: Sünderhauf and Protzel [25] and Olson and Agarwal [17]. Sünderhauf and Protzel [25] propose a robust SLAM back end using "switchable constraints". The central idea is to penalize those loop closure links during graph optimization that deviate from the constraints they suggest between two nodes. Similar to our approach, they change the topological structure of the graph based on identification and rejection of wrong loop closures. In contrast, however, they assess the validity of every loop closure on its own, without forming a general consensus using all the available information. Their method suggests a continuous function governing the state of "switch factors" which may not be appropriate in many cases, for example in traversal paths.

Olson and Agarwal [17] proposed the idea of max-mixtures which can be applied to the problem of loop closure verification. They attach to each possible loop closure a null hypothesis and let the optimizer select the more probable one. The null hypothesis represents a uniform distribution over the whole space and is modelled by a Gaussian distribution with the same mean as the loop closure hypothesis and a very large variance. We show comparisons of the *RRR* algorithm with both these methods in the experimental section.

3 The RRR algorithm

We propose a robust consistency-based loop closure verification method using the pose graph formulation. The RRR algorithm is based on the observation that correct loop closures in conjunction with odometry are very useful in the detection of incorrect loop closures. Our method follows the line of work in which the estimation process itself is used in making the distinction between true and false loop closures. Thanks to the maturity of the SLAM problem, in the last years several very efficient estimation methods have been published as for instance iSAM by Kaess et al. [9], HOG-Man by Grisetti et al. [6], and g2o by Kümmerle et al. [12].

The graph based formulation for SLAM, the so-called "graph-SLAM", models robot poses as nodes in a graph where links from odometry or loop closures form edges or "constraints". Let $\mathbf{x} = (x_1 \ldots x_n)^T$ be a vector of parameters that describes the configuration of the nodes. Let ω_{ij} and Ω_{ij} be the mean and the information of the observation of node j from node i. Given the state $\mathbf{x}$, let the function $f_{ij}(\mathbf{x})$ be a function that calculates the perfect observation according to the current state. The residual r_{ij} can then be calculated as:

$$r_{ij}(\mathbf{x}) = \omega_{ij} - f_{ij}(\mathbf{x}) \tag{1}$$

Constraints can either be introduced by odometry which are sequential constraints ($j = i + 1$), or from place recognition system which are non-sequential. The amount of error introduced by each constraint weighed by its information can be calculated as:

$$d_{ij}(\mathbf{x})^2 = r_{ij}(\mathbf{x})^T \Omega_{ij} r_{ij}(\mathbf{x}) \tag{2}$$

and therefore the overall error, assuming all the constraints to be independent, is given by:

$$D^2(\mathbf{x}) = \sum d_{ij}(\mathbf{x})^2 = \sum r_{ij}(\mathbf{x})^T \Omega_{ij} r_{ij}(\mathbf{x}) \tag{3}$$

The solution to graph-SLAM problem is to find a state $\mathbf{x}^*$ that minimizes the overall error.

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum r_{ij}(\mathbf{x})^T \Omega_{ij} r_{ij}(\mathbf{x}) \tag{4}$$

Iterative approaches such as Gauss-Newton or Levenberg Marquadt can be used to compute the optimal state estimate [12].

We divide the constraints into two sets; S contains sequential links and R contains links from place recognition. Since all constraints are mutually independent, the error in (3) can be written as:

$$D^2(\mathbf{x}) = \sum_{(i,j)\in S} d_{ij}(\mathbf{x})^2 + \sum_{(i,j)\in R} d_{ij}(\mathbf{x})^2 \tag{5}$$

We further divide the set R into n disjoint subsets R_c, where each subset only contains topologically related constraints (sequences of links that relate similar portions of the robot trajectory) such that $R = \cup_{c=1}^{n} R_c$ and $\forall (i \neq j) R_i \cap R_j = \emptyset$. We term each of these subsets as "clusters".

Then the error for set R can be written as:

$$\sum_{(i,j)\in R} d_{ij}(\mathbf{x})^2 = \sum_{c=1}^{n} \sum_{(i,j)\in R_c} d_{ij}(\mathbf{x})^2 = \sum_{c=1}^{n} d_{R_c}(\mathbf{x})^2 \tag{6}$$

where $d_{R_c}(\mathbf{x})^2$ is the error contributed by the cth subset. This simply means that the overall error introduced due to place recognition constraints is the sum of the individual errors of each cluster.

In the absence of any loop closing constraints, the best estimate of the nodes is the one that is constructed from the odometry. If the graph is optimized with just this information, the overall error would be zero because all the constraints agree with each other. Therefore, assuming that if we do not have any outliers in odometry, the error in (3) is caused practically only by the loop closing links. Once we iterate to find the optimal state, the error in the odometry is no longer zero. This increase in odometry error gives us a measure of the metric change that must take place so that the graph conforms to the place recognition constraints. When there are redundant loop closing constraints, this error will be smaller when the corresponding place recognition constraints are correct. This is because a comparatively smaller metric change is needed as opposed to the change required by wrong constraints. Moreover, clusters that suggest the same change would cause smaller errors among them as compared to conflicting clusters. By measuring how much errors the clusters introduce, we can detect which clusters agree with each other. A point to note here is that even though odometry drifts with time, it is still a useful measure of the underlying topology of the graph.

Consider the case when each of these clusters is the only place recognition constraint in the graph. In absence of other clusters, this cluster can make the odometry converge to an acceptable solution, where acceptable means that the overall errors are within some statistical threshold (in this work we use the χ^2 threshold with $\alpha = 0.95$). If this cluster alone deforms the graph enough to make the solution unacceptable, it is highly likely that this cluster is suggesting a wrong place recognition decision. This forms the basis of our intra-cluster test. Mathematically, for any cluster R_i to be individually consistent, the following two conditions must hold:

$$D_G^2(\mathbf{x}) = \sum_{(i,j)\in R_i} r_{ij}(\mathbf{x})^T \Omega_{ij} r_{ij}(\mathbf{x}) + \sum_{(i,j)\in S} d_{ij}(\mathbf{x})^2 < \chi^2_{\alpha,\delta_G} \tag{7}$$

where δ_G are the degrees of freedom of the whole graph. And,

$$D_l^2(\mathbf{x}) = r_{ij}(\mathbf{x})^T \Omega_{ij} r_{ij}(\mathbf{x}) < \chi^2_{\alpha,\delta_l},\ (i,j) \in R_i \tag{8}$$

ensures that if there are any outliers within the cluster they are omitted. δ_l are the degrees of freedom of each constraint. If the criterion in (7) is not met, the whole cluster is rejected. Constraints not meeting the criterion in (8) are removed from the cluster and the rest of the links are accepted as being individually consistent.

Similarly, for a selected subset of clusters C, we term the clusters in C to be jointly consistent if:

$$D_C^2(\mathbf{x}) = \sum_{c=1}^{|C|} \sum_{(i,j)\in R_c} r_{ij}(\mathbf{x})^T \Omega_{ij} r_{ij}(\mathbf{x}) < \chi^2_{\alpha,\delta_C} \tag{9}$$

and

$$D_G^2(\mathbf{x}) = D_C^2(\mathbf{x}) + \sum_{(i,j)\in S} r_{ij}(\mathbf{x})^T \Omega_{ij} r_{ij}(\mathbf{x}) < \chi^2_{\alpha,\delta_G} \tag{10}$$

This first criteria ensures that the present clusters are consistent with each other; the second one ensures the consistency of the clusters with the odometry links.

We also advocate that rather than testing a single hypothesis on its own, clusters are more robust and identify regions of loop closures rather than single pose-to-pose correspondence. This is the case when appearance based place recognition systems generate hypotheses. Having said that, our method does not pose any restriction on the minimum number of links in a cluster and can handle clusters composed of just single links.

An overview of every component of the RRR algorithm is provided in the following section.

3.1 Clustering

Our method starts by collecting topologically related loop closing hypotheses into clusters. Clusters represent sequences of loop closures that relate the same portions of the trajectory. We use a simple incremental way to group them using timestamps. We proceed as follows: with the first loop closure that arrives, we initialize the first cluster R_1. Then, we decide according to (11) if the next loop closure arriving belongs to same cluster or a new cluster needs to be initialized.

$$\omega_{ij} \in R_c \iff \exists \omega_{pq} \in R_c \ \mid\ \|t_i - t_p\| \leq t_g \wedge \|t_j - t_q\| \leq t_g \tag{11}$$

where t_i means the timestamp related to the node i, and t_g can be selected according to the rate at which the place recognition system runs. This threshold defines the cluster neighbourhood.

When the first loop closure (ω_{ij}) arrives in the system, it forms a new cluster. When the second loop closure (ω_{pq}) arrives, both the start (t_p) and the end (t_q) of this loop closures should be within the threshold t_g from the start (t_i) and end (t_j) respectively of some link already present in the cluster, in order to be added to the same cluster. Otherwise, it will form a new cluster containing

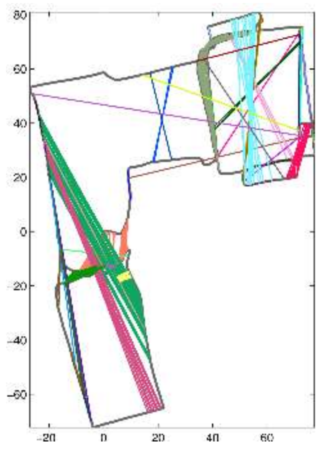


Figure 2: Clustering loop closures. Each cluster is represented by a different colour.

just this link. This allows only loop closures that connect the same parts of trajectory to form clusters. As loop closures arrive in the system, they are incrementally allocated to either existing clusters if they meet the criterion in (11), or form new clusters. This allows the clusters to grow dynamically as long as the loop closures being suggested link similar parts of the trajectory. We assume that the loop closure are directed in time, i.e. loops are closed from the present position to the past ($j < i$).

In our experiments, we consider loop closing hypotheses less than $t_g = 10s$ apart to be a part of the same cluster. A cluster is considered closed if for t_g time no new links are added to it.

An example of clustering can be seen in Fig. 2 where loop closures for one of the sessions of Bicocca dataset [20] are shown. Each cluster is represented by a different colour.

3.2 Intra-Cluster Consistency

After clustering hypotheses together, the next step is to compute the internal consistency for each cluster. This involves optimizing the pose graph with respect to just this single cluster and checking which links satisfy the χ^2 constraint. The links inside the cluster that do not pass this test are removed from the cluster and are no longer considered in the optimization. Algorithm 1 describes the intra-cluster consistency. This procedure is carried out for each cluster and in an incremental implementation it can be performed as soon as a cluster is closed.

3.3 Inter-Cluster Consistency

Having established intra-cluster consistency, we now look for clusters that are mutually consistent. We initially assume that all the clusters that pass the intra-consistency test are also jointly consistent and carry out optimization by including all of them in the optimization problem. Once optimized, we check for any links whose residual error satisfies the χ^2 test. The clusters to which these links belong are then selected and added to the candidate set to be evaluated for joint consistency according to eqs. (9) and (10), see Algorithm 2.

3.4 Best Set Detection

We accept the clusters that are jointly consistent and term them *goodSet*. At this point, we remove the *goodSet* from the optimization and try to re-optimize with the remaining clusters. The

Algorithm 1 Intra_Cluster_Consistency

```
Input: $poses$, $slinks$, $cluster$ of $rlinks$
Output: $cluster$
  add $poses, slinks$ to PoseGraph
  PoseGraphIC $\leftarrow$ PoseGraph
  add $cluster$ to PoseGraphIC
  optimize PoseGraphIC
  if $D_G^2 < \chi^2_{\alpha,\delta_G}$ then
    for each $rlink_l \in cluster$ do
      if $D_l^2 < \chi^2_{\alpha,\delta_l}$ then
        Accept $rlink_l$
      else
        Reject $rlink_l$
      end if
    end for
  else
    Reject $cluster$
  end if
```

Algorithm 2 Inter_Cluster_Consistency

```
Input: $goodSet$, $candidateSet$, PoseGraph
Output: $goodSet$, $rejectSet$
  PoseGraphJC $\leftarrow$ PoseGraph
  add ($goodSet$, $candidateSet$) to PoseGraphJC
  $rejectSet \leftarrow \{\}$
  optimize PoseGraphJC
  if $D_C^2 < \chi^2_{\alpha,\delta_C} \wedge D_G^2 < \chi^2_{\alpha,\delta_G}$ then
    $goodSet \leftarrow \{goodSet, candidateSet\}$
  else
    find the $cluster_i \in candidateSet$ with largest Consistency Index ($D_C^2/\chi^2_{\alpha,\delta_C}$)
    remove $cluster_i$ from $candidateSet$
    $rejectSet \leftarrow cluster_i$
    if $\neg$isempty $candidateSet$ then
      ($goodSet$, $rSet$) $\leftarrow$
      $Inter\_Cluster\_Consistency(goodSet, candidateSet)$
      $rejectSet \leftarrow \{rejectSet, rSet\}$
    end if
  end if
```

Algorithm 3 RRR

```
Input: $poses$, $slinks$, $\mathcal{R}$ set of clusters containing $rlinks$
Output: $goodSet$ of $rlinks$
  add $poses, slinks$ to PoseGraph
  $goodSet \leftarrow \{\}$
  $rejectSet \leftarrow \{\}$
  loop
    PoseGraphPR ← PoseGraph
    $currentSet \leftarrow R \backslash \{goodSet \cup rejectSet\}$
    $candidateSet \leftarrow \{\}$
    add $currentSet$ to PoseGraphPR
    optimize PoseGraphPR
    for each $cluster_i \in currentSet$ do
      if $\exists rlink_j : D_l^2 < \chi^2_{\alpha,\delta_l} \mid rlink_j \in cluster_i$ then
        $candidateSet \leftarrow \{candidateSet, cluster_i\}$
      end if
    end for
    if isempty($candidateSet$) then
      STOP
    else
      $s = goodSet.size$
      $(goodSet, rSet) \leftarrow$
      $Inter\_Cluster\_Consistency(goodSet, candidateSet)$
      if $goodSet.size > s$ then
        $rejectSet \leftarrow \{\}$
      else
        $rejectSet \leftarrow \{rejectSet, rSet\}$
      end if
    end if
  end loop
```

idea behind doing so is that in the absence of the good clusters, other correct clusters will be able to pass the threshold tests. A good analogy would be that of athletes running in a special kind of race in which the observer is only able to see who crosses the finish line first. In order to establish which runner would end up in the second position, we ask the winner to not participate in the race and conduct the race again. We continue asking the winner each time to join the previous winners, to see who will take the next spot. The winners in our case are the member of the *goodSet*. This is shown in Algorithm 2. As long as we keep finding clusters that are jointly consistent with the *goodSet*, it will grow.

An important point to mention here is the use of the *rejectSet*. The reject set contains all the clusters that we checked in the last iteration but found them to be inconsistent with the *goodSet*. We omit them from the optimization process until something is added to the *goodSet*. The main algorithm is given in the Algorithm 3. The algorithm terminates when we are no longer able to find any clusters to add to the *candidateSet*.

3.5 Incremental Realizing, Reversing Recovering (iRRR)

This section presents the incremental version of our RRR algorithm. It is useful in the case when not all the information is available i.e. sequential and loop closing constraints are added as the robot moves. Our algorithm is incremental in terms of clusters, that is to say that as soon as a cluster is closed, the algorithm is triggered to check the validity of the cluster. The incremental algorithm is given in Algorithm 4, which is executed if the current cluster passes the intra cluster consistency test (Algorithm 1).

The main difference to RRR is that rather than now holding the current *goodSet* to be correct and not reconsidering it, all the clusters are considered every time a decision is made. This allows us to change our mind, i.e. accepted clusters can be later rejected and vice versa.

3.6 Multi-session connectivity

Our method is able to deal with multiple sessions in a way similar to “weak links“ and makes the assumption that if there is a correct loop closure between two sessions, this information should align the two sessions correctly after optimization. But this information may or may not be available or may arrive in the system after a long time. Therefore, we need to deal with multiple session in a unified manner, so that if the loop closing information between different sessions becomes available, we can align the sessions, otherwise we maintain up-to-date estimate of all the sessions independently.

The loop closing clusters can be of two types; the first ones are those which close loops within the same session or intra-session loop closures (G_i). The second type are those that connect two sessions with each other or inter-session loop closures (G_{ij}). If we have n sessions that have intra-session loop closures, but no inter-session loops closures, then for these n *un*-connected sessions, the *goodSet* denoted by G^t at time t will be partitionable into n subsets such that $G^t = \cup_{i=1}^{n} G_i^t$ and $\forall i \neq j : G_i^t \cap G_j^t = \emptyset$. G_i^t contains all the good links that belong to session i at time t. It should be noted that this partitioning is based on the sessions to which the loop closing links belong.

Since there are no clusters that link these sets, they are independent of each other, and we can calculate/expand the G_i for each subset independently. Similar partitioning can be done for the *candidateSet*.

If we find a cluster R_{kl} that links session k to session l, this introduces a connection between G_k and G_l via R_{kl}. We fuse these to form a single set. This is given by:

Algorithm 4 iRRR

```
Input: $poses$, $slinks$, $\mathcal{R}$ set of clusters containing $rlinks$, $goodSet_{n-1}$ containing previous $goodSet$
Output: $goodSet$ of $rlinks$
  add $poses, slinks$ to PoseGraph
  $goodSet \leftarrow \{goodSet_{n-1}\}$
  $rejectSet \leftarrow \{\}$
  loop
    PoseGraphPR ← PoseGraph
    $currentSet \leftarrow R \backslash \{rejectSet\}$
    $candidateSet \leftarrow \{\}$
    add $currentSet$ to PoseGraphPR
    optimize PoseGraphPR
    for each $cluster_i \in currentSet$ do
      if $\exists rlink_j : D_l^2 < \chi^2_{\alpha,\delta_l} \mid rlink_j \in cluster_i$ then
        $candidateSet \leftarrow \{candidateSet, cluster_i\}$
      end if
    end for
    if isempty($candidateSet$) then
      STOP
    else
      $s = goodSet.size$
      $candidateSet \leftarrow \{candidateSet, goodSet\}$
      $(goodSet, rSet) \leftarrow Inter\_Cluster\_Consistency(goodSet, candidateSet)$
      if $goodSet.size > s$ then
        $rejectSet \leftarrow \{\}$
      else
        $rejectSet \leftarrow \{rejectSet, rSet\}$
      end if
    end if
  end loop
  $rejectSet \leftarrow \{\}$
```

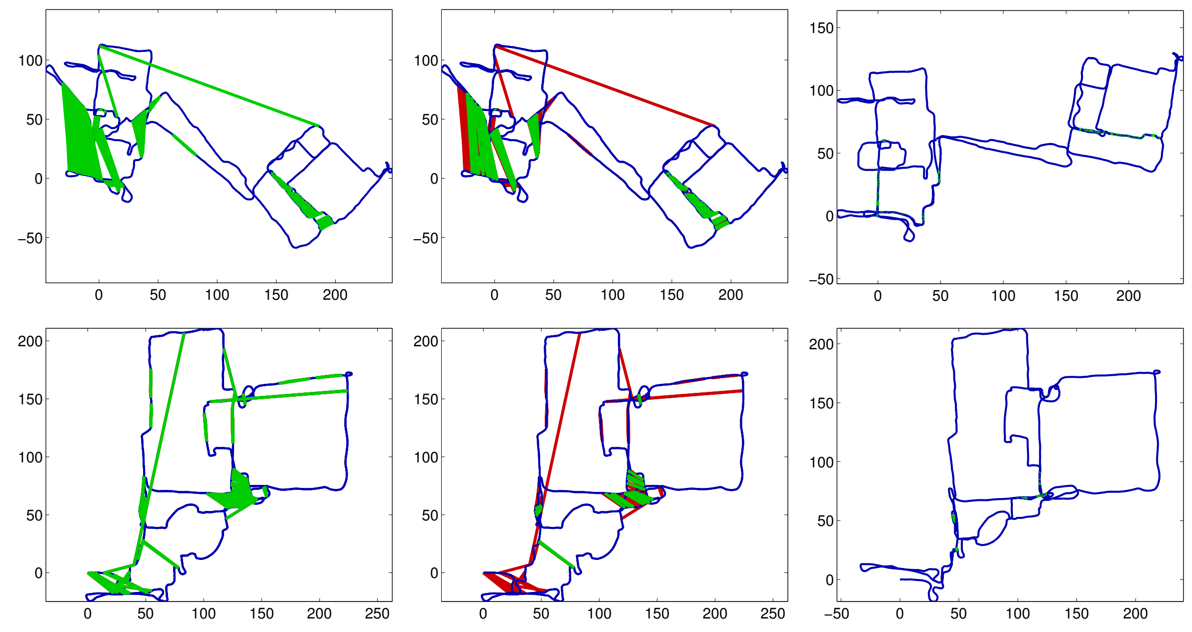

(a) Input from BoW (b) Accepted links (c) Optimized

Figure 3: Results for real datasets: *First row:* Bovisa06 (mixed indoor-out) *Second row:* Bovisa04 (outdoor) **(a)**: Input from the front-end BoW algorithm **(b)**: Links selected (green) and rejected (red) by RRR **(c)**: Optimized graph based on the selected links

$$G_{kl} \leftarrow \{G_k \cup G_l \cup R_{kl}\} \tag{12}$$

Since we have now connected two sessions that were not connected before, the number of unconnected components (the number of partitions of the set n) is reduced by 1. In the ideal case, when all the sessions are connected to each other we get just a single set at the end, indicating that all the sessions are connected either directly or indirectly to all the other sessions. Once we compute/update the new partitioning, the union of these new subsets forms our estimate of the correct loop closures, G^{t+1}.

Consider the case when the robot starts moving in the first session. All the links are intra-sessions links and we only maintain a single subset G_0. After some time, the front-end indicates the start of a new session. Now the intra-sessions clusters will form a new set G_1. As soon as there is a cluster $R_{1,0}$ that links the two sessions, we can merge them to form a single connected graph. If such a cluster is never found, the individual sets G_0 and G_1 maintain the *goodSet* for each individual session.

4 Experiments

In the section we present a set of experiments that show the performance of our algorithm both in batch as well as in incremental mode. We compare against the two other state-of-the-art approaches to robust back-end methods, namely Switchable constraints (SC)[25] and Max-Mixtures (MM)[17]. We explore the effect of variable number of outliers on the three algorithms. Results are also presented for our method under the effect of varying odometry error. Finally results are presented for the incremental version that illustrate recovery from incorrect decisions as well as the the ability of our method to reason over multiple sessions.

Experiments are carried out on real datasets from the RAWSEEDS project [20] such as Bicocca (indoor) and Bovisa (mixed indoor and outdoor), the NewCollege dataset [24], as well as the Intel and Lincoln [5] datasets. We also present results for the synthetic city10000 dataset. For all the experiments with RRR, the back end used was g2o configured with Gauss-Newton and every optimization was carried out for 4 iterations. The code used for SC and MM is available from the respective authors' websites.

4.1 RRR

In the first set of experiments, we qualitatively show the performance of RRR for Bovisa06 and Bovisa04 datasets. The odometry comes from laser scan-matching and loop closures are proposed by a BoW front-end with minimum confidence parameter (α^-) set to 0.15. The results are given in Fig. 3.

This experiment demonstrates the ability of RRR to distinguish between correct and incorrect loop closures. The incorrect loop closures are marked in red and are not considered in the calculation of the optimized graph. It can be seen in Fig. 3(c) that all the loop closures selected are correct since the trajectory, as suggested by odometry, is preserved.

An example execution of the RRR algorithm can be seen in the accompanying multimedia.[1] The video first depicts the odometry (white) gathered from laser scan matching and all loop closures (blue) as suggested by BoW with minimum confidence parameter (α^-) set to 0.0, which generates a lot of false positives. Alternating clusters are represented with alternating colors. It is followed by intra-cluster consistency check. The links that pass this test are shown in cyan and failed ones disappear. For some clusters, links are kept even though some of them disappear. This shows that links within the clusters can be rejected without the whole clusters being rejected. Around the 0:28 mark, a large self-consistent but topologically incorrect cluster also passes the test. Finally an inter-cluster consistency test is carried out and the final optimized map is shown.

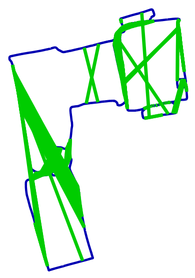
4.2 Comparisons - Single run

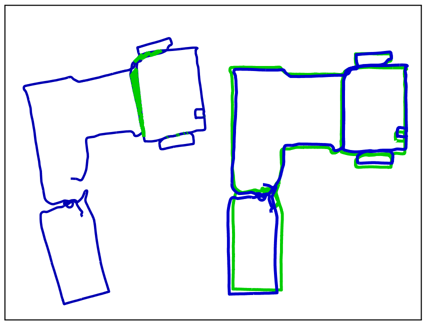
In this experiment we compare the performance of SC, MM and RRR in batch mode on a single run from the Bicocca dataset. The loop closures come from a simple place recognition algorithm (BoW) with the minimum confidence parameter (α^-) set to 0.15. The laser odometry along with the loop closures suggested by BoW are shown in Fig. 4(a).

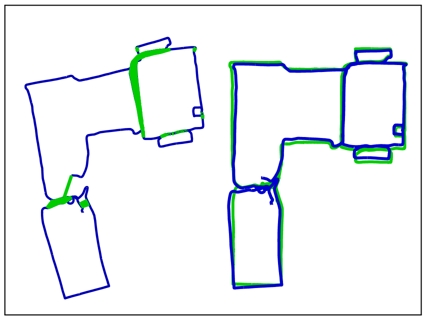
For this experiment, we compare the Absolute Trajectory Error (ATE) for the three algorithms. SC was used with parameters given in the code. In order to correctly determine the weight (w) and scaling (s) parameter for MM, a parameters sweep was carried out in the range $10^{-19} \leq \text{s} \leq 0.1$ and $10^{-9} \leq \text{w} \leq 0.1$. The parameters that provided the least Absolute Trajectory Error were selected for comparison (s=10^{-6}, w=10^{-3}). Both SC and MM were run until convergence. The results are presented in Fig. 4 and Table 1.

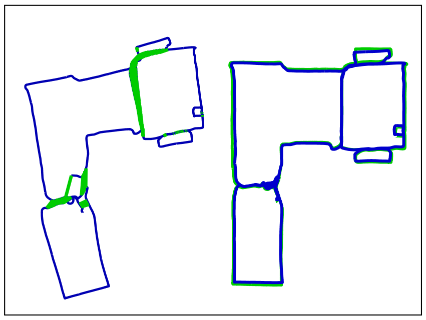
It can be seen from Fig. 4(e) that RRR outperforms the other algorithms in this case. Table 1 provides details of the ATE and its spread as well as the execution time for each algorithm. MM, while being the fastest, is the one which suffers more from drift in odometry for large loops. RRR on the other hand is the most expensive, time-wise, but generates the best results in terms of the overall ATE. This behaviour will be seen again for RRR in cases where there are a few correct loop closures. While the difference between the slowest and the fastest algorithm (MM and RRR)

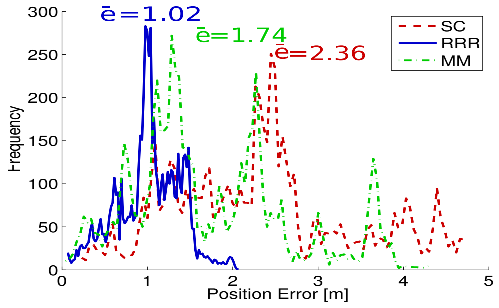
[1]http://www.youtube.com/watch?v=DcCwl4aOkn4


(a) Input.


(b) Output of SC


(c) Output of MM


(d) Output of RRR



(e) Absolute trajectory errors distributions.

Figure 4: **(a):** One of the sessions from Bicocca campus shown in Fig. 11(a)(bottom) with laser odometry and the constraints from the place recognition system. **(b)**: The result of Sunderhauf et. al [25], **(c)**: Olson et. al [17], **(d)**: Output of RRR *left:* the accepted loop closures *right:* optimized graph (blue) and ground truth (green). **(e)**: Distribution of the Absolute Trajectory Errors (ATE) for the final pose graphs against the ground truth (b,c,d:right).

	Mean ATE (m)	Std. Dev ATE (m).	Time (sec)
SC	2.358	1.059	0.88
MM	1.7353	0.923	**0.54**
RRR	**1.018**	**0.367**	8.45

Table 1: Summary of ATE and running time for Fig. 4.

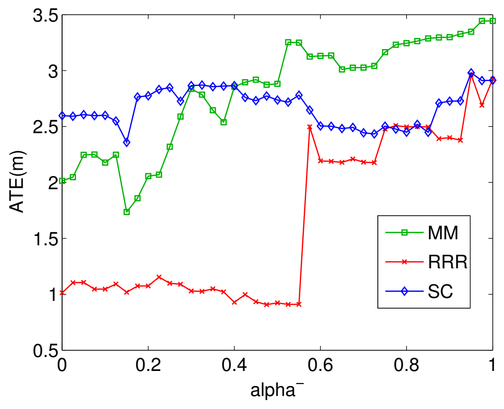
appears to be one order of magnitude, considering that the trajectory was collected over half an hour, the execution time allows for near-real time use of the RRR algorithm.

In order to investigate the robustness of each algorithm to false positive loop closures we compare how each algorithm behaves under varying amount of false positive loop closures in the next experiment.

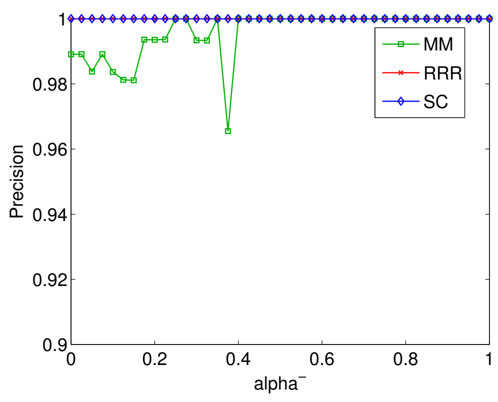
4.3 Robustness to false positive loop closures

In the experiment, we compare the robustness of the RRR, SC and MM to varying amount of outliers in loop closures. The minimum confidence parameter for the front end place recognition system was varied from 0 to 1 with increment of 0.025 to generate 41 different sets of loop closing constraints. The number of loop closures suggested vary from 446 to 23. The number of outliers and inliers that are present in the output from BoW are shown in Fig. 5(d).

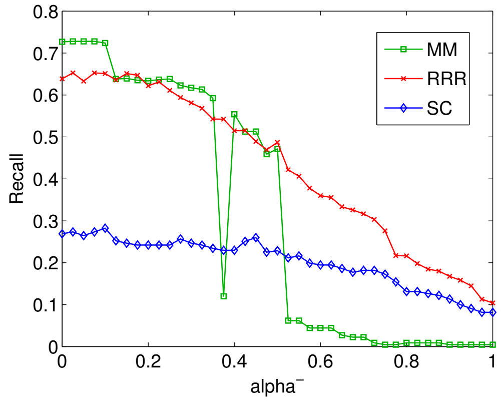
SC was run with the default parameters as provided in the source code. MM was run with the



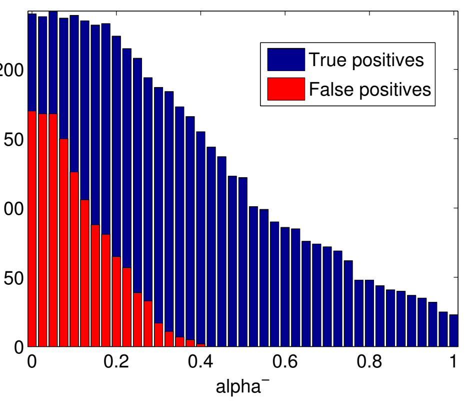
(a) ATE (m)



(b) Precision



(c) Recall



(d) inliers and outliers for BoW

Figure 5: Comparison of RRR, SC and MM: Performance under varying number of outliers

parameters that gave the least ATE for the previous experiment (s=10^{-6}, w=10^{-3}).

The performance of the algorithms is compared based on three metrics: Precision, Recall and Absolute Trajectory Error. The results are shown in Fig. 5. SC does not make a binary decision about the correctness of a loop closing constraint but gives a value between 0 and 1. Therefore, for calculating the precision-recall curves any switch with a value greater than 0.5 was considered to be "on" and "off" otherwise. For MM, all the links in which the suggested loop closing constraint was more probable than the corresponding null hypothesis were selected and included in the precision-recall calculation.

The important metric to look at in the context of robust back-ends is precision. We would like to use an algorithm that provides full precision with a considerable recall. Full precision means that the algorithm never gets confused about which loop closures are correct. Secondly, we would like to have full precision with as much recall as possible. That means that while we only select the correct loop closures, we selected as many of them as possible (ideally, all of them).

As can be seen in the Fig. 5(b) both SC and RRR are able to provide full precision for all the 41 experiments. RRR has a greater recall which amounts to lower ATE. MM is confused by the presence of a lot of outliers but as the number of outliers decreases, MM is also able to achieve full precision.

MM initially has larger ATE because of accepting some false positives. But when it reaches full precision, it suffers from low recall. The lowest point in the ATE plot for MM is the point for which

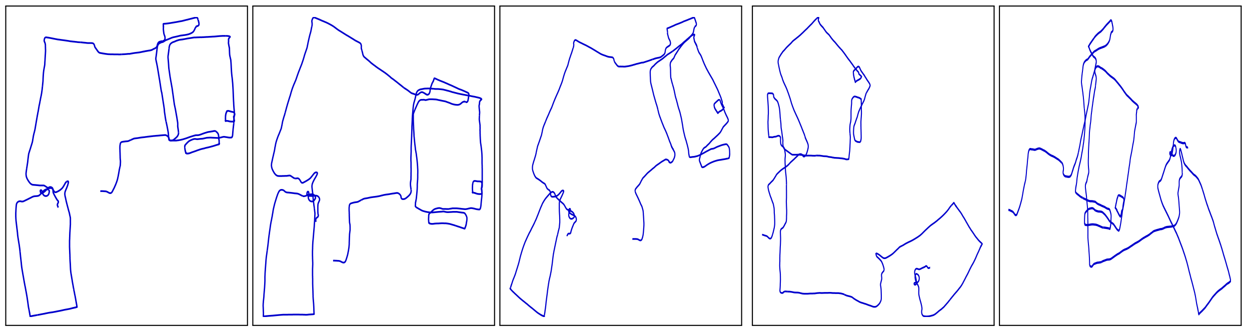
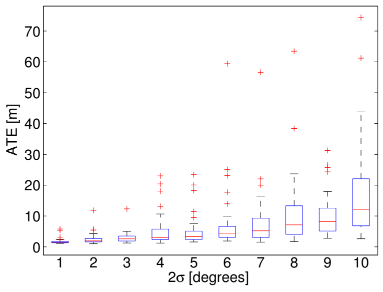
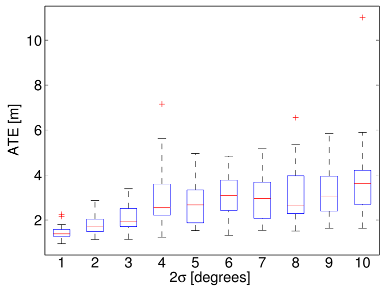


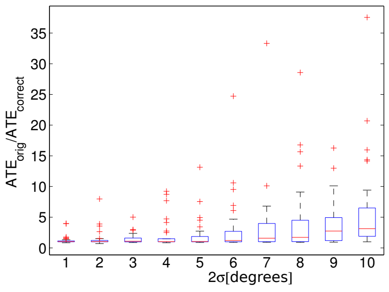
Figure 6: Examples of odometry corrupted by noise : $2\sigma = 2.0,4.0,6.0,8.0,10.0$ degrees



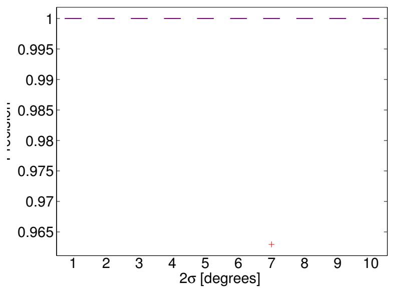
(a) ATE_{orig}



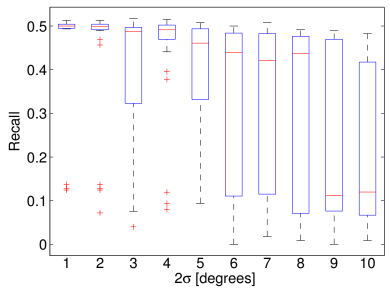
(b) $\text{ATE}_{correct}$



(c) ratio of (a) and (b)



(d) Precision



(e) Recall

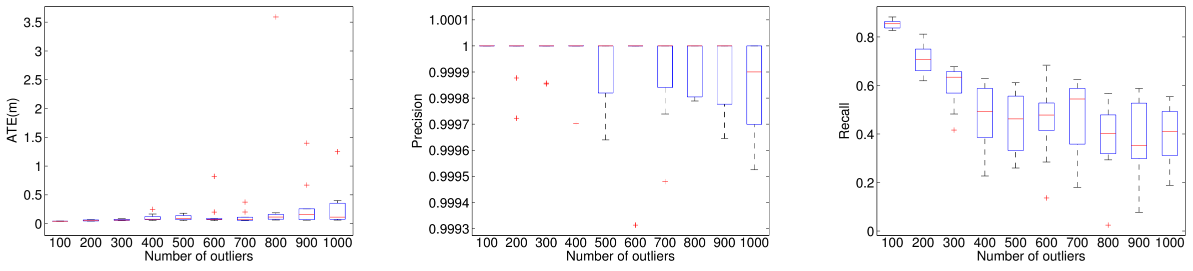
Figure 7: Effects of Odometry noise on RRR **(a)**: ATE_{orig}, ATE for the noisy trajectory with the loop closures determined by RRR. **(b)** : $\text{ATE}_{correct}$, the noisy trajectory but with all the correct loop closures included to calculate the map estimate.

we calculated the parameters using a parameter sweep. This means that if we could calculate the correct parameters for every experiment, MM may perform better. On the other hand, this also means that MM is very sensitive to the tuning parameters and that they do not depend on the trajectory, but on the distribution of the loop closing hypotheses.

4.4 Robustness to odometry noise

Another interesting aspect to look at is the performance of the algorithm with varying odometry error. The main metric of concern is precision, which is the purpose of robust back-end algorithms.

This experiment investigates the performance of RRR under varying amounts of odometry noise. We use the trajectory and loop closings given in Fig. 4(a) and add noise to orientation of



(a) ATE in meters (b) Precision (c) Recall

Figure 8: Results for 100 experiments on the city10000 dataset with varying number of incorrect loop closures

every pose. This has a cumulative effect on the overall trajectory. We simulate 10 noise levels with noise varying from $2\sigma = 1$ degree to 10 degrees. For each noise level, 100 random experiments were generated and RRR was run on them. Fig. 6 illustrates some of the corrupted trajectories.

The results are shown in Fig. 7. In this setup, error can come from two sources. One is the error in odometry and other is the error that might come from false positive or false negative loop closures. In order to have an estimate of how the algorithm behaves under noise, we compare its performance against the case with no noise, i.e. for every output that we get from a noisy version of the odometry (noisy odometry + uncertain loop closures), we also calculate the ATE for the same trajectory but with correct loop closures from a non-noisy version of the experiment (noisy odometry + correct loop closures). This gives us a measure of the amount of performance loss due to not correctly recognizing true positive loop closures.

Fig. 7(a) gives box plots for ATE of the experiments. It can be seen that the ATE is affected by the amount of noise: greater noise leads to greater error. Fig. 7(b) provides the ATE if all of the *correct* loop closures are included to calculate the map estimate. Fig. 7(c) is the ratio of (a) and (b). The medians are marked by the line inside every box. Performance degradation is slow and graceful. We are still able to perform with full precision although recall is affected. In only one of the 1000 experiments, we accept a single false positive loop closure. Recall degrades with varying amount of noise.

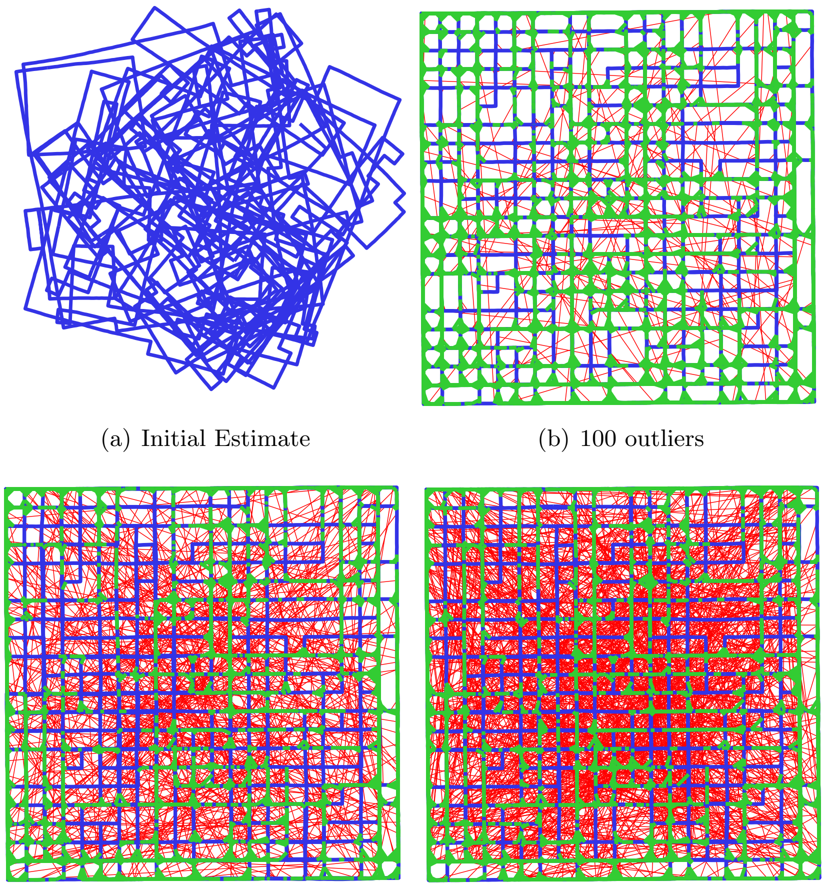
4.5 Synthetic dataset: city10000

While real datasets provide us with a way of evaluating the performance of our algorithm when there are a few loop closings, synthetic datasets can be used to evaluate the performance in the presence of dense loop closing hypotheses.

In the experiment, we evaluate the performance of our algorithm on a synthetic dataset: city10000 provided with iSAM [9]. The dataset simulates a run in a city with $10,000$ nodes and provides 10688 ground truth loop closings.

In order to evaluate the performance of our algorithm under varying number of false positives, randomly generated incorrect loop closures were added to the dataset starting from 100 to 1000 with a step of 100. For each level of outliers, 10 random experiments were generated and the RRR algorithm was run on them. The metrics calculated are ATE, precision and recall. For this dataset, the number of iterations for the optimizer were set to 5. Sample results are shown in Fig. 9 and results for calculated metric are given in Fig. 8.

Fig. 8(b) gives the precision. Since links were added at random, some of them are actually



(a) Initial Estimate (b) 100 outliers

(c) 500 outliers (d) 1000 outliers

Figure 9: Sample results for city10000. **(a)**: Trajectory used for initialization **(b-d)**: Optimized graph after RRR in presence of 100, 500 and 1000 outliers. Accepted links in green, final optimized trajectory in blue and rejected links in red.

outliers	ATE (m)		Precision		Recall	
	mean	median	mean	median	mean	median
100	0.04	0.04	1.00	1.00	0.85	0.85
200	0.05	0.05	1.00	1.00	0.70	0.70
300	0.06	0.06	1.00	1.00	0.59	0.63
400	0.10	0.07	1.00	1.00	0.47	0.49
500	0.10	0.08	0.99	1.00	0.44	0.46
600	0.16	0.07	0.99	1.00	0.45	0.47
700	0.11	0.06	0.99	1.00	0.46	0.54
800	0.46	0.11	0.99	1.00	0.37	0.40
900	0.31	0.15	0.99	1.00	0.37	0.35
1000	0.26	0.11	0.99	0.99	0.38	0.41

Table 2: ATE, Precision, Recall for city10,000

correct but do not exist in the ground truth. This is reflected in the corresponding ATE. If the loop accepted were actually false positives, this would severely corrupt the map, which is not the case

outliers	Total time (sec)		Number of Clusters		time/cluster (sec)	
	mean	median	mean	median	mean	median
100	764.2	785.7	1661.3	1661.5	0.46	0.47
200	648.2	619.8	1757.4	1758.0	0.36	0.35
300	705.8	671.1	1858.3	1859.0	0.37	0.36
400	701.3	699.1	1954.0	1954.5	0.35	0.35
500	765.0	755.5	2053.2	2053.0	0.37	0.36
600	857.8	834.6	2150.5	2150.0	0.39	0.38
700	922.1	918.6	2246.6	2246.5	0.41	0.40
800	978.3	941.5	2343.0	2343.5	0.41	0.40
900	1213.7	1126.5	2441.1	2441.5	0.49	0.46
1000	1362.2	1261.2	2538.5	2540.5	0.53	0.49

Table 3: Execution time RRR: city10,000

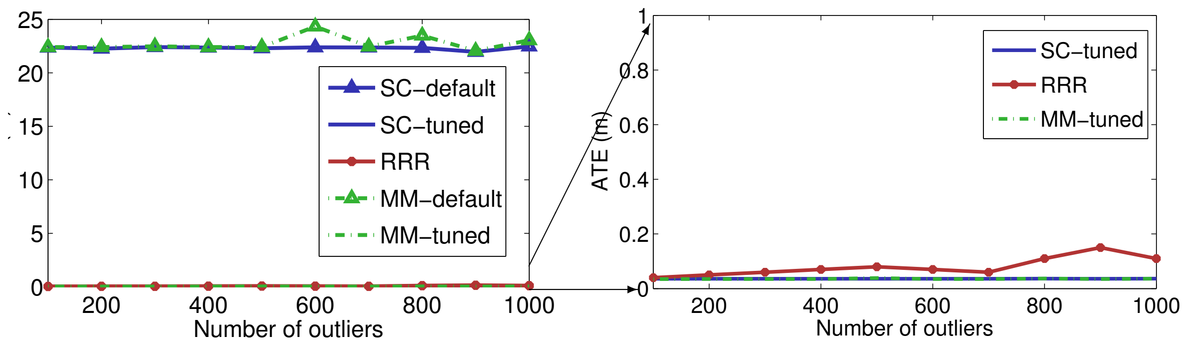


Figure 10: Comparison of ATE for RRR with SC (default,tuned) and MM (default,tuned): **Left:** Using the default paramaters for SC and MM results in rejection of all the loop closures. The solution in that case is just the odometry as shown in 9(a). **Right:** Zoomed version of left. For RRR, the error increases due to decrease in recall with the maximum error at 15cm for a trajectory of 10km.

as can be seen from the corresponding value of ATE in Fig. 8(a).

Recall is affected by the number of outliers as can be seen. Table 2 gives the numerical results for the shown plots. One point to note is that while the median recall at worst is about 41%, the corresponding value of ATE, 0.11 m is very small for the size of the experiment. It means that although we are discarding a lot of correct loop closures, we are not discarding anything important. The maps estimate is still very accurate.

The details for execution time are given in Table 3. The algorithm takes a considerable time to generate the result. The table also gives the number of clusters in each case and time per cluster.

SC and MM were also run on the same 100 datasets generated from the city10000 dataset. As mentioned before, the dataset comes from iSAM and has been widely used in the community to evaluate SLAM back-ends. iSAM expects the information matrix for the constraints to be in the square root form. On the other hand, g2o expects the information matrix to be in the squared form. While it may not have a large effect when comparing the performance of different optimizers, this information is important when deciding which loop closures are correct. In order to illustrate this, we use the correct information matrix (as needed by g2o) and evaluate SC and MM with the default parameters. For SC, the default parameters for the variance on the switch prior is 1 as given in the open-source implementation. For MM, the open-source code provides city10000 corrupted with 1000 incorrect loop-closures. The parameters used there for the scale (s) is 0.01 and weight (w) is 10^{-11}. We use these default parameters and evaluate both of these algorithms on the 100 datasets generated. The results are given in Fig. 10. Both SC and MM fail to converge and reject all loop closures. This leads to zero recall and a very high trajectory error.

In a second set of experiments we searched for a set of tuning parameters that would enable the algorithms to converge to an acceptable solution. For SC, we found that setting the variance on the switch prior to 0.1 gave the best results. Similarly, for MM we did a parameter sweep and the parameters s=.1 and w=10^{-5} gave the best results. The corresponding values for trajectory error are shown in Fig. 10.

The previous experiment illustrates the dependence of both of these algorithms on the tuning parameters. It is possible to tune these parameters in a batch context or when some prior information about the magnitudes of false positives is available, but the final goal is to make these methods work on a robot in an incremental fashion. It is not clear how these parameters can be automatically tuned for different problems.

On the synthetic dataset, properly tuned SC and MM perform at full precision and recall, outperforming RRR. The first thing to note is that compared to real datasets, synthetic datasets have a considerably larger number of loop closures. When using statistical tests with such large number of variables, the discriminative ability of the test suffers. This might lead to loop closures that have small errors to be accepted as being correct. At the same time, this affects the over all consistency and leads to rejection of other loop closures, hence the reduced recall.

4.6 Long term operation

In this experiment we illustrate the incremental version of our RRR algorithm and how it can be used to connect multiple runs. The main difference from the batch version is that in the incremental implementation, odometry and loop closures constraints are added one by one. The algorithm is triggered as soon as a cluster is closed.

The interesting aspect of solving the problem incrementally is that it allows us to change our mind, i.e. if at one point something was considered correct and later evidence suggests otherwise, the constraints can be removed. Similarly, if we find support for something that was previously rejected, it can be reincorporated into the optimization, with the result of better map estimate. Also, we illustrate the ability of our algorithm to reason over the alignment of different sessions based on identification of correct loop closures between them.

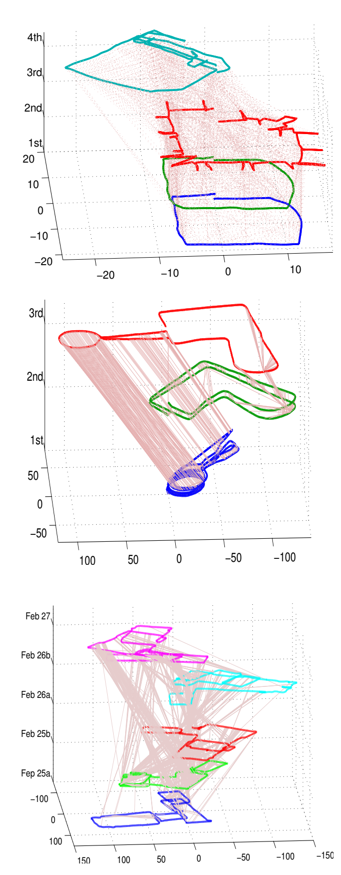
In this experiment we show the output of running the incremental RRR (iRRR) algorithm on three different datasets; Intel, New College and Bicocca.

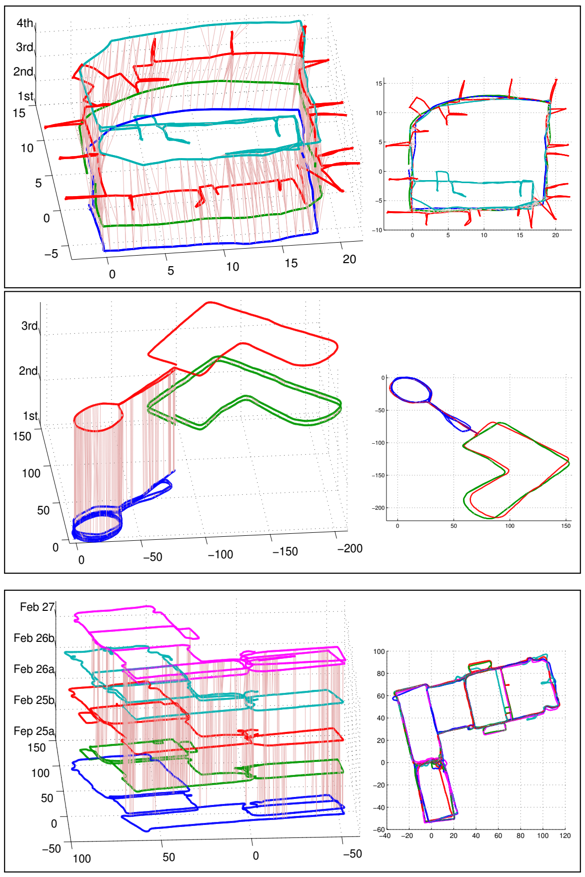
Loop closing links for New College and Bicocca were calculated using BoW, while random loop closings were added to the Intel dataset. The Bicocca dataset consists of 5 runs in an indoor environment, while Intel and New College were divided to into four and three sessions respectively, to simulate odometry black-outs.

The results are shown in Fig. 11. iRRR is able to recover the correct loop closures as well as the correct transformation between the different sessions. iRRR is able to reject incorrect loop closures and align all the sessions correctly. Fig. 11(c) provides the computations time for making a decision.

A video showing the execution of iRRR for the Bicocca dataset can be seen in the attached multimedia[2]. It shows a multisession scenario, in which loop closures are accepted (green) or tested and rejected (red). Near the end of the video, a cluster that was previously rejected is again accepted when evidence supporting it becomes available.

[2]http://youtu.be/aL0Vd0CIMrM


(a) Inputs.


(b) Output of our iRRR algorithm.

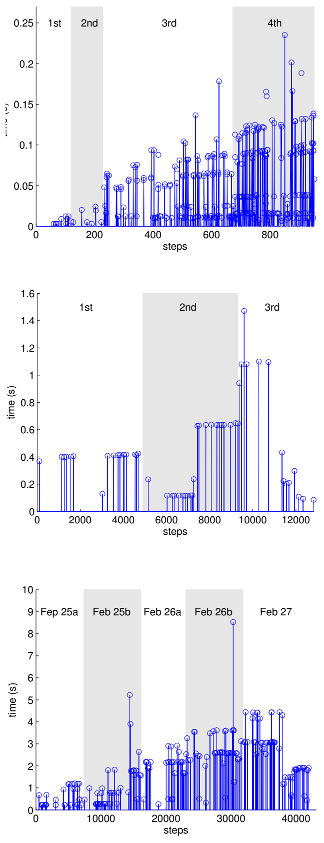
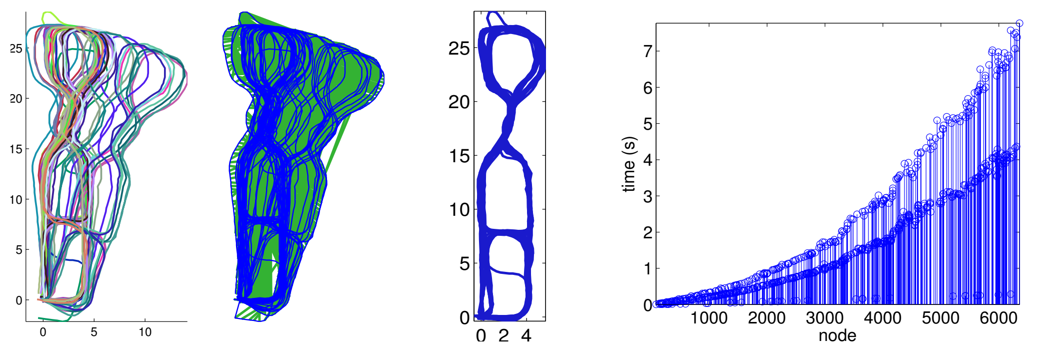

(c) Computational time.

Figure 11: Multi-session experiments. **(a)**: The input; odometry (each session is a different color and height) and loop closures (pink). **(b)**: The output of our proposal, each session on different height (left) and the floor-view (right). On top, the Intel dataset divided by us into four sessions and corrupted by 600 wrong loop closures. In the middle, the NewCollege dataset using visual odometry divided into three sessions and with a BoW+gc place recognition system. Bottom, the Bicocca multi-session experiment from the RAWSEEDS project, odometry from laser scan matching and the loop closures from a BoW+gc place recognition system. Each session is in its own frame of reference. **(c)**: The computational time of our proposed method against the step when is triggered.

We also found a very good example of long term navigation in the Lincoln dataset [5], 36 sessions on a changing environment of an office floor at the University of Lincoln. We run iRRR on this multi-session data. This dataset provides omnidirectional images and laser scans. We compute the odometry by simple scan matching of the laser returns, see Fig. 12(a), and the loop closure constraints taking the best match comparing the GIST descriptors of the Omnidirectional images as described in [21], see Fig. 12(b). The final result with our algorithm is shown in Fig. 12(c).

The final graph contains 6357 nodes with 6123 loop closure constraints grouped in 668 clusters, from which iRRR accepts 4495 links. The algorithm takes 7.92s in the final step to reconsider all the information in the 36 sessions gathered during seven weeks.



(a) Odometry. (b) loop closings. (c) iRRR. (d) Time per decision against node where cluster is closed.

Figure 12: Multi-session experiments. **(a)** The input; laser odometry (each session is a different color). **(b)** Loop closing hypotheses generated by selecting the closest match based on the GIST descriptor **(c)** Floor view of the optimized graph after iRRR on the 36 sessions. Notice all the sessions have been aligned properly. **(d)**Time per decision at the node where the cluster is closed.

5 Discussion

The RRR algorithm is a consistency-based approach for a robust backend of a SLAM system. Our method is agnostic to the type of sensor used for place recognition as well as to the front end place recognition algorithm. All that our method requires is a sequential constraint generator (normally laser/visual or other form of odometry) and a place recognition system that is able to provide non-sequential links. The iRRR algorithm is an incremental version that can additionally be used to connect multiple session based only on the correct loop closures identified between them.

We have presented the comparisons of our method against two recently proposed approaches to robust back ends, namely Olson's Max-mixtures and Sünderhauf's switchable constraints. On real datasets where loop closing constraints are sparse, our method is able to perform at full precision with a considerable recall, resulting in a smaller absolute trajectory error. The aim of SLAM is not just to build a nice looking map but to use it further to carry out higher level tasks such as planning [3] and navigation [26]. In that regard, loop closing links provide traversability information. Switchable constraints may provide a good map estimate but the switches are governed by a continuous function, where as loop closures should either be completely accepted or completely rejected. In certain cases, this might cause problems when the map is actually being used to carry out higher level tasks. Max-mixtures is a mathematically sound approach but it requires a great deal of parameter tuning, as has been shown in comparisons. Both SC and MM are fast and very useful when there are dense loop closing hypotheses such as in synthetic datasets. However, in real world scenarios loop closing are comparatively sparse and there is weak evidence for MM and SC to work on. In those cases, RRR outperforms both. In synthetic datasets RRR also achieves good results. While execution time is greater, it is still real time for the duration of the experiments. For the RRR algorithm as well as iRRR, the time complexity is $O(n)$ where n is the number of clusters being considered. The total execution time depends on the underlying graph structure and thus the corresponding optimization time.

Both RRR and iRRR algorithms can reverse decisions. In order to preseve the current graph estimate, the validity of each new decision is first evaluated in a backup graph. In the batch mode, intra-cluster as well as inter-cluster evaluations are carried out on a backup of the current graph estimate. When the calculation of the goodSet is carried out, the corresponding optimized graph is also calculated. The current graph estimate is then updated with this new estimate. Similarly, for the incremental implementation, every new decision is taken on a backup of the graph and the current graph estimate is always maintained to be up to date with the current goodSet.

With respect to the effect of odometry error on our algorithm, even in the presence of a considerable amount of noise, our method is able to work at full precision, which is a highly desirable quality for a robust back-end. We have presented results for the incremental version of the algorithm and demonstrate its ability for incremental reasoning and determining correctly the alignment between sessions based on the available evidence.

The proposed algorithm includes two parameters. The first one is the threshold that controls the size of the clusters, t_g. For real dataset, this threshold t_g can be determined by the rate at which the odometry and place recognition systems work. In general, every place recognition algorithm ignores part of the recent trajectory while trying to find the place recognition hypotheses. The value of t_g can be set to half of the duration for which the place recognition ignores the previous poses. For all the experiments with real datasets, t_g was set to 10 seconds. For city10000 we assume that the place recognition rate and odometry rate are the same, in effect ignoring the past 10 seconds of trajectory. The second parameter is the number of iterations for the intra- and inter-cluster consistency test. In datasets with sparse loop closure hypotheses, which is the case with real world datasets or a real robot, the number of iteration can be set to 3. For large datasets with a great number of loop closures, this parameters can be set to 5. We could not find any difference, rather than a greater execution time, by setting the parameter any higher. This parameter can be thought of as having three values, $3, 4, 5$ for sparse, medium and dense loop closings in the graph. This can be set according to the permissiveness of the front end place recognition algorithm.

With regards to the failure modes, unmodeled errors, such as incorrect information on the odometry or loop closure links, are by far the worst enemy of the algorithm. Moreover, the algorithm is based on the idea that incorrect loops can be rejected when new contradictory evidence becomes available. This assumption may not always be true. In regions where loop closures might be sparse and far separated, it will be difficult to detect failures.

6 Conclusion

In this work we have proposed a new algorithm to deal with the problem of incorrect loop closures introduced into the map estimate by the front-end place recognition algorithm. Our method correctly identifies false positives and removes them to robustly recover the correct map estimate. We have shown a batch as well as an incremental version of the algorithm and investigated the effect of number of outliers and odometry error on the method. We demonstrated that our method can work in real scenarios where there are sparse loop closing hypotheses as well as in synthetic datasets with dense hypotheses. Our method is more robust than the alternative state-of-the-art methods, although for a higher computational cost. Additionally, our method is able to reason on spatially connected or unconnected sessions and maintains up to date information and aligns them as soon as correct linkages can be determined between any two sessions.

Future work includes relaxing the assumptions on odometry so that the algorithm can consider outlier measurements also in odometry. Currently, the most expensive operation is the calculation of intra-cluster consistency. One good tool to resolve this would be spectral clustering [16]. This would enable efficient distinction between correct and incorrect links within the cluster. We are also working on extending the current approach to work in dynamic environments. Over long time periods, the structure of the environment may change because objects may be added or removed.

Currently our method does not cater for such changes but rather accumulates all the information over the whole time. Have all the information and related loop closures, reasoning can be done for representing the belief about the current structure of the environment. Thirdly, the current time complexity prevents the algorithm from working over extended time periods (months) as the time to make a decision grows linearly with the number of clusters. We are working on deciding some grouping of clusters based on Euclidean distances in the optimized graph in order to reduce the number of clusters present in the system at a given time. Also, we plan to explore the application of our method with the reduced pose graph formulations of Johannsson et al. [8] or Kretzschmar et al. [11] to reduce the computational load for very long operation.

Acknowledgements

The authors would like to thank Niko Sünderhauf for the vertigo library and Pratik Agarwal, and Edwin Oslon for the source code of max-mixtures.

References

[1] G. Agamennoni, J.I. Nieto, and E.M. Nebot. An outlier-robust kalman filter. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1551 –1558, may 2011. doi: 10.1109/ICRA.2011.5979605.

[2] C. Cadena, D. Gálvez-López, J.D. Tardós, and J. Neira. Robust place recognition with stereo sequences. *IEEE Transaction on RObotics*, 28(4):871 –885, 2012. ISSN 1552-3098. doi: DOI: 10.1109/TRO.2012.2189497.

[3] H. Carrillo, Y. Latif, J. Neira, and J.A. Castellanos. Fast minimum uncertainty search on a graph map representation. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2504–2511, 2012. doi: 10.1109/IROS.2012.6385927.

[4] M. Cummins and P. Newman. Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, 2010. doi: 10.1177/0278364910385483. URL `http://ijr.sagepub.com/content/early/2010/11/11/0278364910385483.abstract`.

[5] F. Dayoub, G. Cielniak, and T. Duckett. Long-term experiments with an adaptive spherical view representation for navigation in changing environments. *Robotics and Autonomous Systems*, 59(5):285–295, 2011.

[6] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg. Hierarchical optimization on manifolds for online 2d and 3d mapping. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 273 –278, may 2010. doi: 10.1109/ROBOT.2010.5509407.

[7] P.J. Huber. Robust regression: asymptotics, conjectures and monte carlo. *The Annals of Statistics*, 1(5):799–821, 1973.

[8] H. Johannsson, M. Kaess, M.F. Fallon, and J.J. Leonard. Temporally scalable visual SLAM using a reduced pose graph. In *RSS Workshop on Long-term Operation of Autonomous Robotic Systems in Changing Environments*, Sydney, Australia, Jul 2012.

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

[10] K. Konolige and J. Bowman. Towards lifelong visual maps. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1156–1163. IEEE, 2009.

[11] H. Kretzschmar, C. Stachniss, and G. Grisetti. Efficient information-theoretic graph pruning for graph-based slam with laser range finders. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 865 –871, sept. 2011. doi: 10.1109/IROS.2011.6094414.

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

[13] Y. Latif, C. Cadena, and J. Neira. Realizing, Reversing, Recovering: Incremental Robust Loop Closing over time using the iRRR algorithm. In *Proc. IEEE/RJS Int. Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, October 2012.

[14] Y. Latif, C. Cadena, and J. Neira. Robust Loop Closing Over Time. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.

[15] J. McDonald, M. Kaess, C. Cadena, J. Neira, and J.J. Leonard. Real-time 6-DOF multi-session visual SLAM over large scale environments. *Robotics and Autonomous Systems*, (0):–, 2012. ISSN 0921-8890. doi: 10.1016/j.robot.2012.08.008. URL `http://www.sciencedirect.com/science/article/pii/S0921889012001406?v=s5`.

[16] E. Olson. Recognizing places using spectrally clustered local matches. *Robotics and Autonomous Systems*, 57(12):1157–1172, December 2009.

[17] 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.

[18] E. Olson, J. Strom, R. Morton, A. Richardson, P. Ranganathan, R. Goeddel, M. Bulic, J. Crossman, and B. Marinier. Progress towards multi-robot reconnaissance and the MAGIC 2010 competition. *Journal of Field Robotics*, 29(5):762–792, 2012.

[19] A. Ranganathan and F. Dellaert. Online probabilistic topological mapping. *The International Journal of Robotics Research*, 30(6):755–771, May 2011. doi: 10.1177/0278364910393287. URL `http://ijr.sagepub.com/content/early/2011/01/23/0278364910393287.abstract`.

[20] RAWSEEDS. Robotics advancement through Webpublishing of sensorial and elaborated extensive data sets (project FP6-IST-045144), 2009. http://www.rawseeds.org/rs/datasets.

[21] A. Rituerto, A. C. Murillo, and J. J. Guerrero. Semantic labeling for indoor topological mapping using a wearable catadioptric system. *to appear in Robotics and Autonomous Systems, special issue Semantic Perception, Mapping and Exploration*, 2012.

[22] G. Sibley, C. Mei, I. Reid, and P. Newman. Vast-scale outdoor navigation using adaptive relative bundle adjustment. *The International Journal of Robotics Research*, 29(8):958–980, 2010.

[23] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, October 2003.

[24] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman. The new college vision and laser data set. *The International Journal of Robotics Research*, 28(5):595–599, May 2009. ISSN 0278-3649. doi: http://dx.doi.org/10.1177/0278364909103911. URL `http://www.robots.ox.ac.uk/NewCollegeData/`.

[25] N. Sünderhauf and P. Protzel. Switchable Constraints for Robust Pose Graph SLAM. In *Proc. IEEE/RJS Int. Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, 2012.

[26] R. Valencia, J. Andrade-Cetto, and J. M. Porta. Path planning in belief space with Pose SLAM. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 78–83. IEEE, 2011.

[27] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard. Dynamic Pose Graph SLAM: Long-term Mapping in Low Dynamic Environments. In *Proc. IEEE/RJS Int. Conference on Intelligent Robots and Systems*, 2012.