Robust Loop Closing Over time for Pose Graph SLAM

主要贡献

1、能够区分loop closure 是否正确；correct or incorrect
2、不正确的loop closure约束从pose-graph中移除
3、将RRR算法扩展为增量的方式，便于计算
4、可以统一的方式处理多场景

RRR algorithm

Realizing, Reversing, and Recovering

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.

Key idea is that of consensus: correct loop closures tend to agree among themselves and with the sequential constraints, while incorrect ones do not.

序列约束

Input: the sequential pose constraints, and a place recognition system for the non-consecutive loop closure constraints.
The output : an optimized pose graph

The RRR algorithm

节点：机器人姿态

边： 约束（里程计以及闭环）

$\mathbf{x} = (x_1 \ldots x_n)^T$ 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

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

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.

第一步、Clustering

We use a simple incremental way to group them using timestamps.

第二步、Intra-Cluster Consistency

对每个cluster内部进行一致性检验，不符合的约束将删除

第三步、Inter-Cluster Consistency

check for any links whose residual error satisfies the $\chi 2$ test.

第四步、通过上述检验的goodset 移除之后重新检验

idea behind doing so is that in the absence of the good clusters, other correct clusters will be able to pass the threshold tests.

Algorithm 1 Intra_Cluster_Consistency

Input: *poses*, *slinks*, *cluster* of *rlinks*
Output: *cluster*

```
add poses, slinks to PoseGraph
PoseGraphIC ← PoseGraph
add cluster to PoseGraphIC
optimize PoseGraphIC
if D_G^2 < χ^2_{α,δ_G} then
    for each rlink_l ∈ cluster do
        if D_l^2 < χ^2_{α,δ_l} then
            Accept rlink_l
        else
            Reject rlink_l
        end if
    end for
else
    Reject cluster
end if
```

Algorithm 2 Inter_Cluster_Consistency

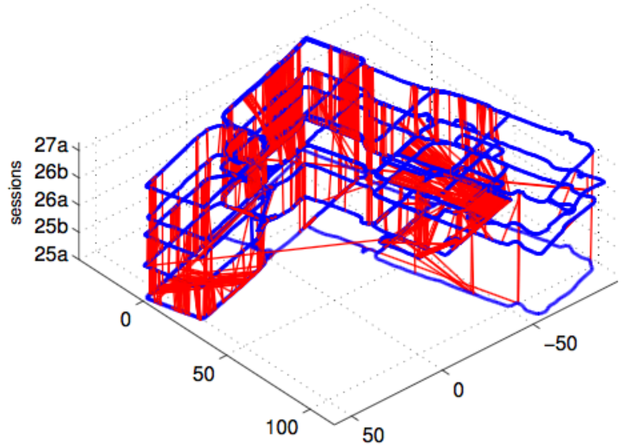
Input: *goodSet*, *candidateSet*, PoseGraph
Output: *goodSet*, *rejectSet*

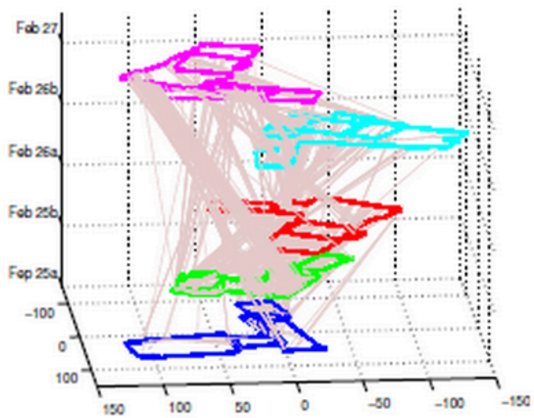
```
PoseGraphJC ← PoseGraph
add (goodSet, candidateSet) to PoseGraphJC
rejectSet ← {}
optimize PoseGraphJC
if D_C^2 < χ^2_{α,δ_C} ∧ D_G^2 < χ^2_{α,δ_G} then
    goodSet ← {goodSet, candidateSet}
else
    find the cluster_i ∈ candidateSet with largest Consistency Index (D_C^2/χ^2_{α,δ_C})
    remove cluster_i from candidateSet
    rejectSet ← cluster_i
    if ¬isempty candidateSet then
        (goodSet, rSet) ←
        Inter_Cluster_Consistency(goodSet, candidateSet)
        rejectSet ← {rejectSet, rSet}
    end if
end if
```

Algorithm 3 RRR

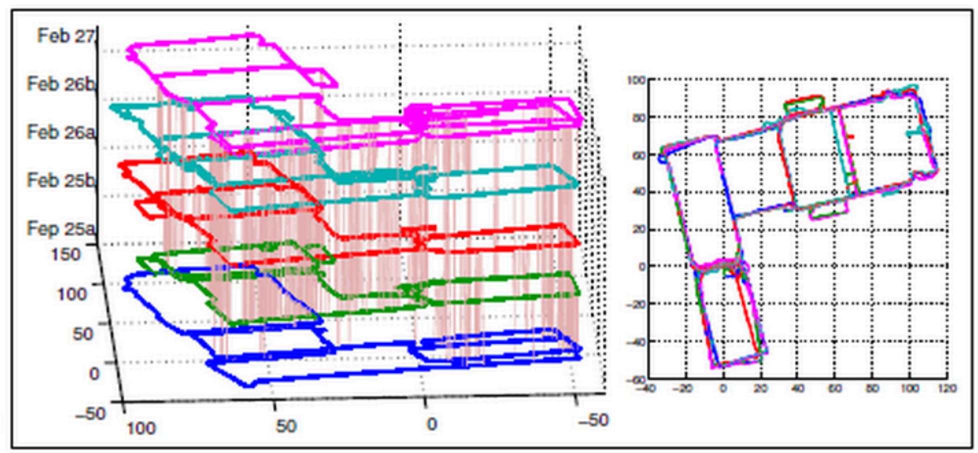
Input: *poses*, *slinks*, $\mathcal{R}$ set of clusters containing *rlinks*
Output: *goodSet* of *rlinks*

```
add poses, slinks to PoseGraph
goodSet ← {}
rejectSet ← {}
loop
  PoseGraphPR ← PoseGraph
  currentSet ← R\{goodSet ∪ rejectSet}
  candidateSet ← {}
  add currentSet to PoseGraphPR
  optimize PoseGraphPR
  for each cluster_i ∈ currentSet do
    if ∃rlink_j : D_l^2 < χ^2_{α,δ_l} | rlink_j ∈ cluster_i then
      candidateSet ← {candidateSet, cluster_i}
    end if
  end for
  if isempty(candidateSet) then
    STOP
  else
    s = goodSet.size
    (goodSet, rSet) ←
    Inter_Cluster_Consistency(goodSet, candidateSet)
    if goodSet.size > s then
      rejectSet ← {}
    else
      rejectSet ← {rejectSet, rSet}
    end if
  end if
end loop
```





(a) Inputs.



(b) Output of our iRRR algorithm.