This supplementary material is organized in two sections. In Section I we provide details on how group outliers are generated and present an in-depth analysis of the experimental results for group outliers. In Section II we perform parameter study on the effect of the penalty scaling for *pro-odometry* clusters.

I. Effect of outliers' structure

Originated from Sunderharf's work [1], research on outlier mitigation [2], [3], [4] has commonly injected two types of outliers into clean datasets for performance validation: random outliers, where singular outliers are generated by randomly choosing a pose-pose pair and no inner structure among outliers are enforced[1]; and group outliers, where a specific inner structure is enforced on the outliers so that within one group, all the edges are consistent with each other topologically. We also used the script from [1] to generate random singular outliers. However, the proposed group outlier generation scheme has three fundamental issues:

- It produces all outliers in groups. This is equivalent of assuming all outliers during the whole trajectory are generated from perceptual aliasing instances that produce mutually-consistent outliers, ignoring random singular outliers entirely.
- All groups have the same number of outliers. This is equivalent of specifying all the perceptual aliasing instances to span the same number of poses.
- With each edge referred as $(from, to)$, the $from$ poses of the edges in one group are in strict consequential order. So are the to poses. An example is shown in Fig.1a.

Based on our observations, these three conditions are rarely true in reality. Group outliers exist together with random singular outliers most of the time, and individual perceptual aliasing instances natually produce different number of mutually-consistent outliers. To improve and simulate perceptual aliasing in a more realistic way, we design a new scheme for group outliers generation, shown in Algorithm 1. Instead of constraining the number of loop closures in all groups to be one constant, our method produces a mixture of singular outliers and group outliers of different sizes. In addition, instead of strictly consequential edges, we randomly generate the pose-pose pair within a specific range and inject a noise level of ± 1 pose. The differences between group outliers from [1] and our methods are illustrated in Fig. 1.

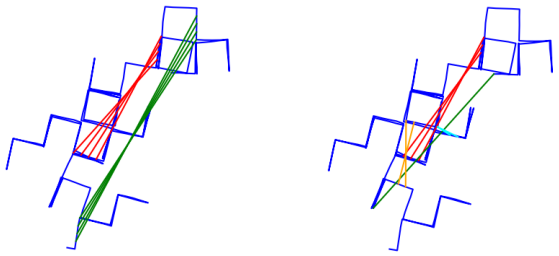
With our carefully designed group outlier generation schemes, we conduct experiments on datasets with 10%, 20%, 30%, 40%, 50% outliers. Experimental results on CSAIL are shown in Fig. 2. OFCC performed much worse than other methods by accepting more outliers. SC ($\sigma_s = 1$), DCS and CPS had stable performance regardless of the number of outliers. And CPS achieved the lowest ATE by

[1]Note that inner structure can still form, since one randomly generated singular outlier can be consistent with some other randomly generated singular outliers. The emphasis here is that it is not enforced.

Algorithm 1: group outlier generation

Input: The maximal group range max_range, the total number of loop closures, n_loops
Output: Outlier list

```
loop_counter = 0
while loop_counter < n_loops do
    Randomly generate a number a ∈ [1, max_range]
    if a == 1 then
        Randomly generate a singular loop closure and
          append to the outlier list
        loop_counter++
    else
        Randomly generate a number b ∈ [2, a]
        Randomly generate a pose-pose pair (x, y) as
          the center of this perceptual aliasing instance
        counter = 0
        while counter < b do
            Randomly draw a pose x' from a uniform
              distributiontion of which the mean is x,
              and bounded by [x − a/2, x + a/2]
            Determine pose y' from
              y' = x' − x + y + random[−1, 1]
            Append [x', y'] into the outlier list
            counter++
        loop_counter = loop_counter + b
```



(a) group outliers from [1] (b) group outliers from our method

Fig. 1: Comparison between different group outlier generation methods: blue lines indicate the trajectory, other lines indicate the injected outliers with each different color representing a different groups of outliers

rejecting most outliers and also accepting most inliers. For DCS, different Φ value didn't affect the ATE significantly. SC ($\sigma_s = 0.1$) performed better than SC ($\sigma_s = 1$) when the number of outliers is low, but soon deteriorated by accepting more outliers as the number of outliers is higher than 30%.

Experimental results on the Manhattan datasets are shown in Fig. 3. Both OFCC and SC ($\sigma_s = 0.1$) failed due to accepting outliers even at 10% outliers. And in doing so, OFCC rejected a lot more inliers then other methods. Both DCS ($\Phi = 1$) and DCS ($\Phi = 5$) performed reasonably when the number of outliers were low, but soon deteriorated as the number of outliers increased. SC ($\sigma_s = 1$) achived the lowest ATE on this configuration, rejecting all outliers while still accepting most of the inliers. CPS performed as well as SC ($\sigma_s = 1$) at low number of outliers, and its performance slowly deteriorated as the number of outliers increased. We note that this is the only configuration in our experimental setups that CPS performed slightly worse than another method, although we can still observe that it avoided accepting loop closures that would cause large global

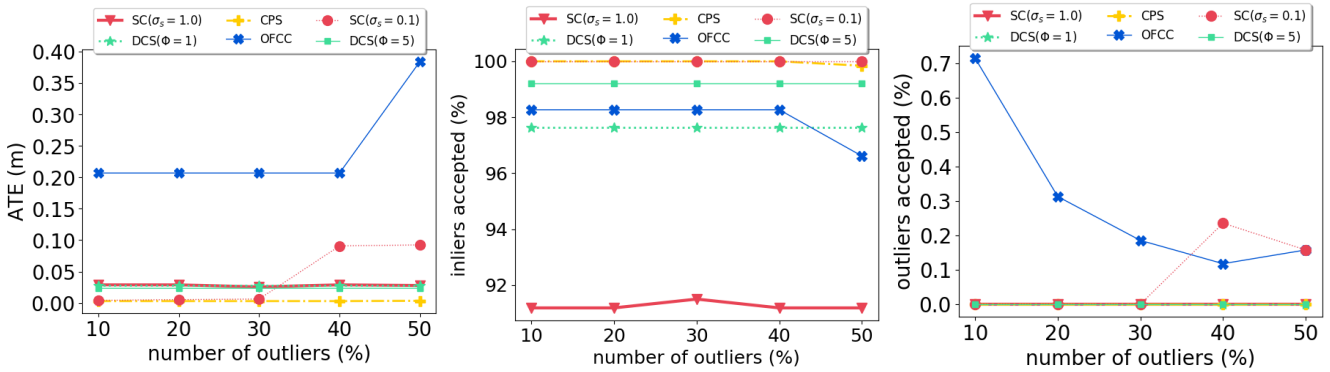


Fig. 2: Comparison of different outlier mitigation methods on CSAIL with group outliers

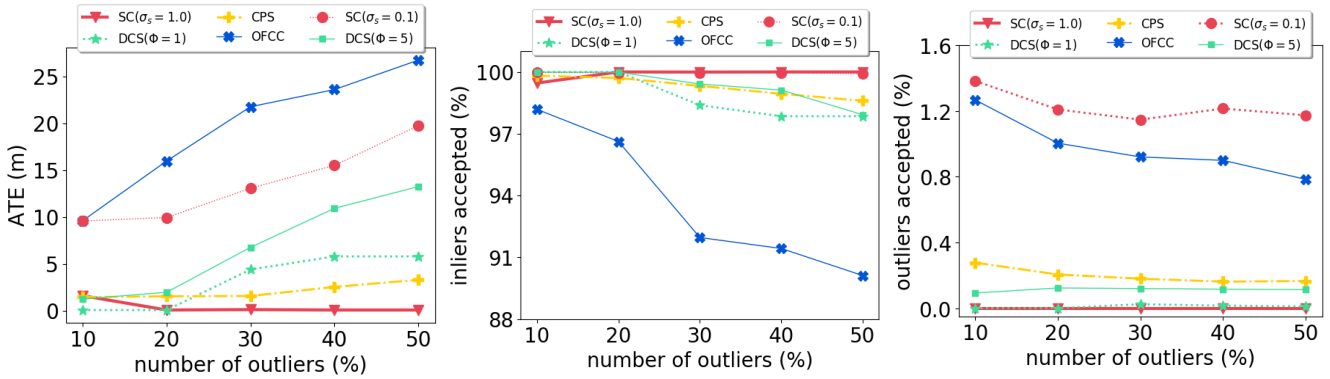


Fig. 3: Comparison of different outlier mitigation methods on Manhattan with group outliers

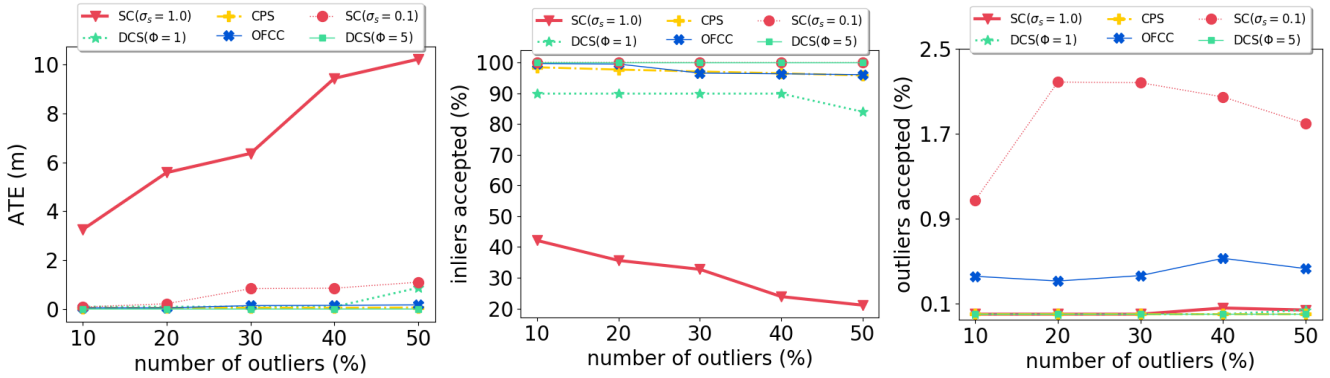


Fig. 4: Comparison of different outlier mitigation methods on Intel with group outliers

distortion: comparing its performance and DCS, both DCS ($\Phi = 1$) and DCS ($\Phi = 5$) accepted less outliers than CPS, but had much higher ATE.

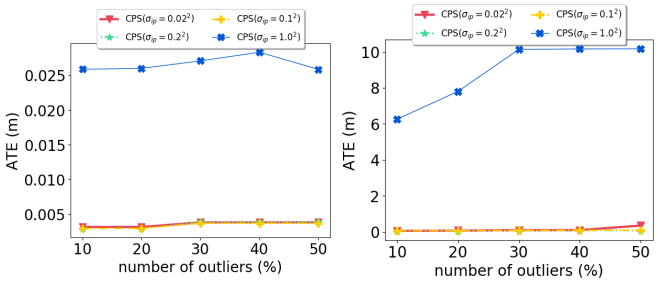
Experimental results for the Intel dataset are shown in Fig.4. SC ($\sigma_s = 1$) failed by rejecting too many inliers while SC($\sigma_s = 0.1$) performed significantly better by making good compromise. DCS ($\Phi = 1$) performed well untill the number of outliers is at 50%, while DCS ($\Phi = 5$) is not affected. OFCC and CPS performed consistently well regardless of the number of outliers.

Comparing the experimental results presented here and those for random singular outliers, we note that the outliers' structure affects the performance of state-of-the-art methods in various ways: OFCC performed significantly worse on CSAIL with group outliers than with random outliers. SC ($\sigma_s = 0.1$) performed significantly differently for the group outliers and random outliers on the CSAIL dataset, making parameter tuning even more difficult. Our proposed method not only performed better or comparable with other methods at their best parameters, but also exhibited similar behaviors for outliers with different structures. Its performance deteriorated gracefully as the number of outliers increased.

II. Effect of Penalty Scaling on pro-odometry cluster

As explained in the paper, we treat the *pro-odometry* clusters differently by decreasing $\sqrt{\sigma_i}$ by a factor 10. Here we conduct an ablation study on this strategy to examine its usefulness. We refer to this parameter as σ_{ip} here to differentiate it from the variance of the prior constraints for *against-odometry* clusters, which is set to 1.0 at all times.

We vary the parameter value: $\sigma_{ip} \in [0.02^2, 0.1^2, 0.2^2, 1.0^2]$, to examine its effect on performance. When $\sigma_{ip} = 1.0^2$, it is equivalent to disabling the scaling strategy for *pro-odometry* clusters since the same variance is used for *against-odometry* clusters. Experimental results on the CSAIL and Intel datasets are presented in Fig. 5. It is apparent that when the scaling strategy is disabled, the ATE significantly increases on both datasets, and the effect is much more severe for the Intel dataset, due to its noisy odometry. When the strategy is enabled and with varying parameter values, the ATE changes by a negligible amount, with the only exception being $\sigma_{ip} = 0.02^2$ for the Intel dataset. For this configuration, the highest ATE is still below 1m. For the Bicocca dataset, the ATE increases from 1.102m to 2.905m when the scaling strategy is disabled, and reduced to 0.661m when $\sigma_{ip} = 0.2^2$, 0.605m when $\sigma_{ip} = 0.02^2$.



(a) Results on the CSAIL dataset

(b) Results on Intel dataset

Fig. 5: Parameter study on the penalty scaling of *pro-odometry* cluster

In summary, we can conclude that treating *pro-odometry* and *against-odometry* clusters differently with penalty scaling played a significant role in our method. And despite having one extra parameter for the *pro-odometry* clusters, the performance of our method is only minimally affected when a wide range of parameter values are used. It is significantly less sensitive to parameter tuning compared to other methods.

References

[1] N. Sunderhauf and P. Protzel, "Switchable constraints for robust pose graph SLAM," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1879–1884, 2012.

[2] P.-Y. Lajoie, S. Hu, G. Beltrame, and L. Carlone, "Modeling Perceptual Aliasing in SLAM via Discrete-Continuous Graphical Models," pp. 1–13, 2018. [Online]. Available: http://arxiv.org/abs/1810.11692

[3] P. Agarwal, G. D. Tipaldi, and L. Spinello, "Robot Map Optimization using Dynamic Covariance Scaling," *2013 IEEE International Conference onRobotics and Automation (ICRA)*, 2013.

[4] L. Carlone, A. Censi, and F. Dellaert, "Selecting good measurements via $\ell 1$ relaxation: A convex approach for robust estimation over graphs," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2667–2674, 2014.