updates. In our current implementation we perform batch optimization. We need to rerun the EM algorithm if new measurements are observed. Incremental implementation of the framework is subject for future work.

C. Incremental Optimization

We introduce the theoretical basis for an incremental implementation of the EM algorithm for the problem. As suggested in [6], if the joint probability is fully factorized with regards to the examples, as is in our case, we can use the following algorithm for update in the E and M step while preserving the correctness:

a) E step: In equation 7, if we have one new observation $\boldsymbol{z}_{K+1}$ at time $K+1$, then

$$\begin{aligned}\boldsymbol{x}^*_{i_{K+1}}, \boldsymbol{l}^*_{j_{K+1}} = \underset{\boldsymbol{x}_{i_{K+1}}, \boldsymbol{l}_{j_{K+1}}}{\arg\max} \{ & \sum_i ||f_i(\boldsymbol{x}_{i-1}, \boldsymbol{u}_i) - \boldsymbol{x}_i)||^2_{\boldsymbol{\Gamma}_i} \\ & + \sum_{k=1}^{K+1} w_{j_k} ||v_k(h_k(\boldsymbol{x}_{i_k}, \boldsymbol{l}_{j_k}) - \boldsymbol{z}_k)||^2_{\boldsymbol{\Sigma}_k} \\ & + \sum_{k=1}^{K+1} ||1 - v_k||^2_{\boldsymbol{\Xi}_k} \}\end{aligned} \tag{10}$$

and equation 10 can also be written in the form of 9, but with less parameters. So, the incremental E step becomes

- Choose one observation term k to be updated, such as $k = K+1$
- Set $\boldsymbol{x}^{(t)}_{i_p} = \boldsymbol{x}^{(t-1)}_{i_p}, \boldsymbol{l}^{(t)}_{j_p} = \boldsymbol{l}^{(t-1)}_{j_p}$ for $p \neq k$
- Set $\boldsymbol{x}_{i_k}, \boldsymbol{l}_{j_k}$ according to equation 7 in equation 9's form.

b) M step: The same step as introduced in section IV-A, using the derivation in equation 6.

With this incremental EM algorithm, the E step can be much faster since the optimization problem has less parameters. Also, after several iterations, we may use the model to update the parameters by adding more observations, without the need to run the EM algorithm on all the observations again.

V. EXPERIMENTS

We implemented our method as discussed previously and compared the result with various alternative approaches. Our implementation is based on the graph optimization framework g2o [5], and we programmed a plug-in type library in C++ to represent our modified objective function while reusing the Gauss-Newton optimization functionality. The type library exposes properties of the edges including the error metric, reweighted information matrix, and robust Mahalanobis distance function.

A. Datasets

The first dataset is the commonly compared landmark-based dataset Victoria Park released with iSAM [4]. We apply different types of simulated corruption to the dataset to generate multiple synthesized datasets to evaluate the effect on different approaches. For the purpose of evaluating performance against moving landmarks, several other commonly used datasets are not suitable because they are pose-only graphs without landmarks. The Victoria Park dataset of 2-D odometry and landmark observations contains 6969 robot poses, 6968 odometry measurements, 151 landmarks, and 3640 landmark measurements obtained.

The second dataset is based on a set of real world data collected in crowded environment at the Alcazar of Seville with a lot of tourists [10]. The original datasets provide wheel odometry, stereo images, and laser scans. In the indoor GPS denied environment, the provided ground truth map and trajectory are built using a non-linear batch optimization-based SLAM method with an approximate accuracy of 20 cm. We extract potentially moving landmarks from a indoor subset of this dataset through a standard pipeline of feature detection and extraction, feature correspondence, and stereo estimation of keypoint depth implemented with OpenCV [2] and ROS [9]. Due to lack of visual detection of loop closure, we also add in a loop closure obtained from laser scans instead of image data at the start-end point. This dataset tests the typical scenario of visual SLAM with wheel odometry. The extracted dataset consists of 4892 robot poses, 5808 measurements, and 13454 measurements.

B. Prior Methods with Moving Landmarks

In the first test, we evaluate the performance of the state-of-the-art robust SLAM method, a implementation of Max-Mixture [7], given dynamic landmark measurements. Max-Mixture is a robust extension to classical graph SLAM using Gaussian mixture in factor representation, that is, $\boldsymbol{x}_i = \sum_c \phi_c \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ for component c and weight ϕ_c, with observations being components, also similarly for $\boldsymbol{z}_i$. Max-Mixture uses a max function to approximate and efficiently evaluate the sum of Gaussians. It is well-known for its capability of handling a large amount of incorrect loop closures, but it is untested against moving landmarks, and this experiment can be representative to other approaches in the robust SLAM literature.

We apply different perturbations to all observations associated with a certain landmark which has the most observations associated in the dataset, and study the different effects of the perturbations on the optimal estimate of the trajectory and the map of all landmarks obtained by Max-Mixture graph SLAM. The Victoria Park dataset contains 2-D odometry and landmark observations. As shown in Figure 3 the left plot is the control group without perturbation, and is accurate to the ground truth. A single southward bias is applied to all observations of the landmark in the middle plot, simulating sensory outliers. A temporally increasing bias is applied to all observations of the landmark in the right plot, simulating a southward moving landmark.

As the middle plot shows, Max-Mixture is still capable of handling noise of large bias or spurious loop closures introduced by simulated sensory fault. It correctly rejects outlier landmark observations and recovers the position of the perturbed landmark. However, it completely fails to