# Data Association as part of SLAM Optimization

Codename: Lost Bumblebee

Bo Xue xuebo@umich.edu
Dingyi Sun dysun@umich.edu
Peter Westra pwestra@umich.edu
Sidhartha Dey siddey@umich.edu
Kaiduo Fang fangkd@umich.edu

***Abstract*— Typically data association is not considered to be a component of the optimization framework when solving SLAM via the method of Incremental Smoothing and Mapping (iSAM). The common way of solving this problem is to make a hard data association for measurements and implement resulting inertial and geometric measurements into the factor graph. The work demonstrated in this paper utilizes semantic measurements to perform probabilistic data associations for the measurements seen by the sensor. This leads to the smoothing of the trajectory and mapping of the landmarks being dependent on the data associations. Thus, the data associations add another degree of freedom within the non-linear least squares problem which allows the solver to reach a more optimal solution. This method was tested on the KITTI dataset and error comparisons from the ground truth trajectory are made between visual odometry, ISAM 2 implementation with hard data associations and iSAM 2 implementation with soft data associations.**

***Index Terms*— SLAM, Semantic, Graph optimization**

## I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) is a mature research area which aimed to extend robot autonomy. When mapping the environment for landmarks, traditional SLAM approaches in the past made use of geometric information, such as corners, edges, etc. However, this metric information provides no conclusive way to say anything meaningful about the landmarks or environment in general semantically.

In the current *robust-perception age* [4][1], visual odometry and semantic methods have become more prevalent in SLAM research and draw upon the decades of advancements in the field of computer vision. On the other way, the resent insight into the structure of the SLAM problem and advancements in the fields of space linear algebra has boosted the graph-based SLAM and currently it becomes the state of the art method with respect to speed and accuracy[7].

### A. Related Work

Our project is motivated by the approach to semantic SLAM presented by Bowman et al. in [2]. Their algorithm combined inertial, geometric and semantic observations into a joint optimization problem and included data association in the optimization framework by leveraging the idea of probabilistic data association. By including the data association in the factor graph, the solution will move towards the trajectory, landmark locations and data associations which lead to the minimization of factor graph residuals. Erroneous data association can lead to divergence in the SLAM algorithm and Bowman et al. show that using semantic along with geometric information can effectively address this as well as loop closure.

### B. Our Contribution

In this paper, we develop a framework leveraging probabilistic data association to tackle semantic SLAM. The full working code can be found on our github page [https://ika-chan.github.io/lost-bumblebee/]. In Section II, we break down our algorithm and discuss some keypoints of each step. Section III elaborates the testing used to validate our algorithm. Section IV and V highlight our findings and logical reasoning of the same.

## II. PROPOSED ALGORITHM

The purpose of this project is to implement a full graph slam with soft data association. The framework of the SLAM algorithm can be separated into independent steps:

1) Extract inertial sensor information from the data-set.
2) Process geometric features from camera data using SURF.
3) Object detection and classification.
4) Integrate the sensor information into a single pose graph.
5) Solve the full pose graph optimization using GTSAM.

After combining the parts, the quantitative results can be compared with the data-set benchmarks. The factor graph representation of the proposed algorithm can be seen in 1. Two types of landmarks will be considered in this factor graph, which are geometric landmarks and semantic landmarks. The geometric landmarks are features extracted by SURF. These feature are tracked in the SLAM framework but they are not been considered for long-term loop closure. For simplicity, we take marginalization of these geometric features from the factor graph used to obtain the relative SE(3) transformations between poses. The semantic landmarks are observed using camera and Lidia. In stead of using the hard association method that set up one factor between the current pose and the observed landmark, the soft association is applied here, which basically add factors to all landmarks been observed at this moment. Each factor are weighted to carry certainty of this association.

### A. Visual Odometry

The means of obtaining relative SE(3) transformations between poses used was monocular visual odometry. This method was implemented in MATLAB and the algorithm can be described as follows;

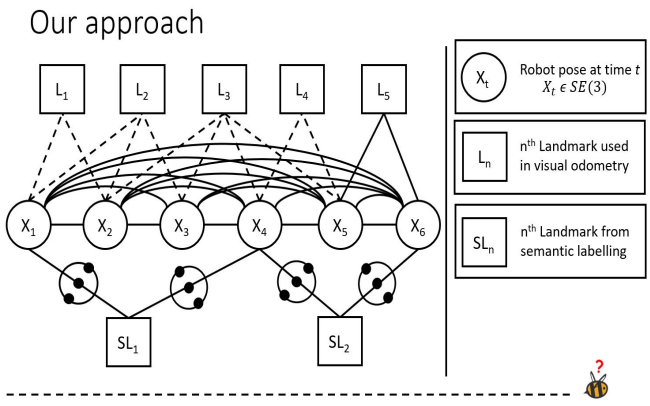


Fig. 1. Algorithm framework

- SURF (a geometric object detection bulit in function) is used to extract geometric features from a camera frame.
- Inliers between frames are tracked with SURF (with outliers being removed by RANSAC).
- The geometric features with known correspondences across several camera frames are used to create relative 3D measurements via triangulation.
- The optimal SE(3) transformation between point clouds is solved and used as the inertial information to be input into the factor graph.

Monocular visual odometry alone is not enough to obtain good tracking of the vehicle becuase it will inherintly drift overtime. With the addition of other sensor information the estimated trajectory should be able to approach the ground truth.

### B. Object Detection

To detect objects from an image, we use the existed network Mask RCNN[8]. Mask RCNN is a flexible and general framework for object instance segmentation which efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. It extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.

The Mask RCNN gives us several bounding boxes' pixel locations in the image, the object's name, and how sure it is of the detection. From these data, we only save the "vehicles" including cars, trucks, bicycles etc. We compute the centroid pixel location from bounding box location, save it and also the certainty. In order to improve the accuracy of output data, we only save the vehicles that Mask RCNN is more than $95\%$ sure of the detection.

Using the pre-trained parameters from COCO dataset, we get every frame's vehicle location by apply the Mask RCNN on each frame. This information is used to initialize landmarks and also to create the probabilistic data associations for landmarks that have already been viewed at a previous pose.

### C. Landmark Location

To get 3D relative location from 2D pixel location. We used Lidar data. The Lidar data is initially a $N$ by $4$ matrix, where $N$ is the number of points the Lidar detected, and each column contains the $X$, $Y$, $Z$ location and a reflectance factor [6]. The $X$, $Y$, $Z$ location is the relative location from Lidar. The $X$ axis is forward, $Y$ axis is leftward, and the $Z$ axis is upward. We project all the 3D Lidar point location onto the image plane, which gives us a look-up table. For each points in the image, we get its 3D location in the way that find the closest point in the look-up table, find the corresponding 3D Lidar point. Because the range of Lidar is about $80m$, we can only get the 3D location that is within $80m$ away. Otherwise the location information will not be accurate.

For all the centroids of detected vehicles, we implement the above way to get its 3D relative location, i.e. the landmark locations. To get the global location, we use these relative location based on the visual odometry information generated by the above method.

### D. Graph Construction and Optimization

The graph construction can be separated into two different components. Those being the semantic factors and the inertial factors. The pipeline of information and implementation is as follows;

- The inertial information is obtained from the monocular visual odometry and used as a constraint between poses. The residual of this contraint can be seen in the following equations.

$$f_i^I(\chi) = -logp(I_{ij}|\chi) \tag{1}$$

$$f_i^I(\chi) = ||r_{I_{ij}}||^2_{\Sigma_{ij}} \tag{2}$$

  The covariance of the odometry measurements is a constant parameter which is tuned based on the observed accuracy of the SE(3) transformations.
- The measurements for the cars observed at the particular pose are obtained from MASK RCNN. The Mahalanobis distance is calculated for each landmark within the camera frame for each measurement. This is done by using the corresponding range measurement given by the relative positional vector and the range given by the landmark location and pose location. Only landmarks with a positive relative X positional vector to the camera pose are tested. The output from this process is a vector of data associations and a matrix with the Mahalahnobis distance for all potentially seen landmarks over all measurements. If the lowest Mahalanobis distance is not below the set threshold the landmark is initialized. If the measurement corresponds to a landmark in the map, constraints will be added between the pose and the landmark, with the number of constraints being equal to the number of cars seen in the particular image.
- The following equations detail the residual of the constraints created by the landmark measurements.

$$f_{kj}^s(\chi, L) = -w_{kj}^{t,(i)} logp(s_k^b|l_j^p) \tag{3}$$

$$f_i^I(\chi) = ||s_k^b - h_\pi(x_t, l_j)||^2_{R_s/w_{kj}^{t,(i)}} \quad (4)$$

- The weight seen in the previous equation is calculated by dividing the corresponding Mahalanobis distance for a specific landmark measurement pair by the sum of the Mahalanobis distances for all the landmarks with associations over that particular measurement and subtracting the value from 1, details in Algorithm 1. This normalizes the weight and causes the landmark measurement pairs with the smaller Mahalanobis distances to have the greatest weights. The noise of the sensor is a static parameter which was tuned to obtain the optimal results.
- With the weights and the covariances for each landmark/measurement pair, N squared constraints are added to the factor graph with N being the number of measurements obtained at that particular time step.

The final optimization framework can be described by the following equation.

$$\hat{x}_{1:T}, \hat{l}_{1:M} = argmin\, \Sigma_{k=1}^{K} \Sigma_{j=1}^{M} f_{kj}^s(\chi, l_{1:M}^p) + \Sigma_{t=1}^{T-1} f_t^I(\chi) \quad (5)$$

## III. Experiment

We applied the proposed technique on sequence number 18 of the KITTI dataset [6][5]. The monocular visual odometry and the MASK RCNN were run offline before the iSAM solver due to the lack of computational resources necessary for an online implementation. The visual odometry, landmark location and iSAM2 solver were implemented in MATLAB while the Mask RCNN was implemented in Python.

The visual odometry calculated the relative SE(3) transformations between each camera frame and the MASK RCNN provided semantic information of the vehicles seen in each camera frame. The monocular visual odometry algorithm was run up to the 2301st camera image in increments of 100 images to reduce the drift and increase the speed of the algorithm. iSAM2, [3][9][10], was chosen to be the back-end because of its ability to reduce the computational complexity of solving large SLAM problems and the flexibility to customize the graph, since there were a lot more soft association factors to be included. The data for the first 2301 frames was used to analyze the difference in trajectory error between the monocular visual odometry, iSAM2 trajectory with hard data associations and iSAM2 trajectory with soft data associations. The results from this analysis can be seen in the following section.

## IV. Results

The relative transformations obtained from the use of monocular visual odometry were used to plot a trajectory vs. the ground truth vehicle pose. This plot can be seen in Figure 2.

The results of estimated trajectory from the monocular visual odometry are good, while compared with the main stream method presented in KITTI benchmark, the result is not competitive. One of the reasons may be that the algorithm was tracking the camera pose without applying any loop closure strategies to reduce the drift in the result. The quantitative error of the trajectory estimate provided by the monocular visual odometry can be in Table I.

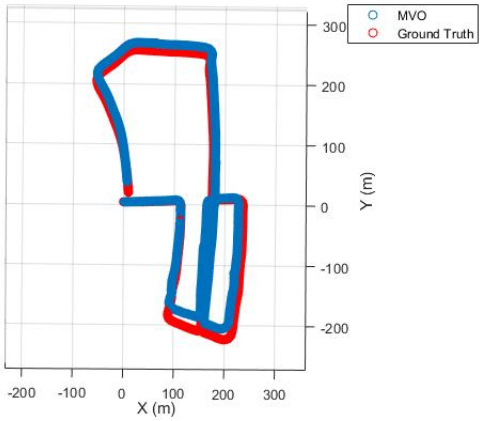


Fig. 2. Monocular Visual Odometry Results

TABLE I
Mean SE(3) error on KITTI sequence 18

| Method | SE(3) error (1/m) |
|---|---|
| Monocular Visual Odometry | 17.0363 |

The output from the MASK RCNN can be seen in Figure 3. The issue is that because we use the pre-trained parameters from COCO dataset [11] instead of KITTI, sometimes it comes up with some wrong detections. For example, some object maybe detected as vehicles although they are actually not. And some vehicles are detected into several pieces or just part of the vehicle is detected, like the front truck shown in Figure 3.

As described in the previous sections, the LIDAR data is used to create relative 3D measurements for all of the vehicles in the camera frames. The visual depiction of this process can be seen in Figure 4. Here also comes with an issue that some points within the range of Lidar are not detected by the Lidar, like some points in the window areas of vehicles. This happens because the Lidar does not receive the reflected rays from these areas, like some window area shown in Figure 4. This makes the landmark locations not accurate because we get the 3D location based on the detected Lidar points.

The relative measurements from the monocular visual odometry and MASK RCNN processing were input into the factor graph computed in iSAM2 and the results can be seen in Figure 5. The red data points are the ground truth of poses, blue are the result using the soft data association algorithm and the orange data points are the result of the hard associations. The same results are depicted in Figure 6 with ground truth poses in red, iSAM2 result in black and the seen landmarks being displayed in green. The analytical results for this Localization solution can be seen in Table II

Fig. 3. Output from MASK RCNN image processing

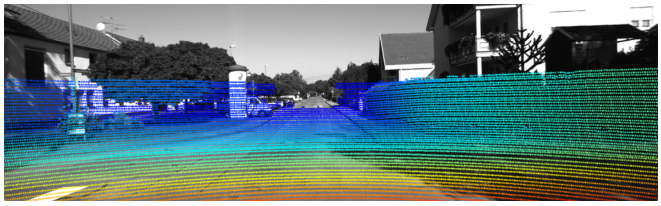

Fig. 4. Depiction of LIDAR 3D points

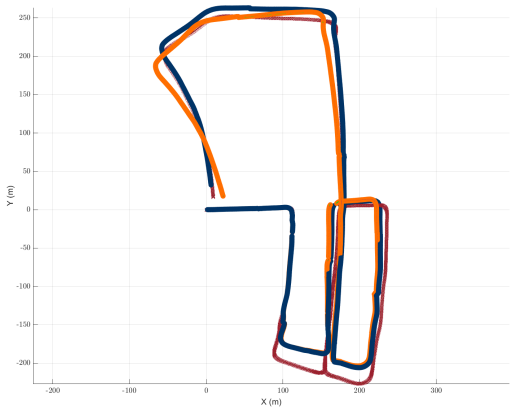


Fig. 5. Trajectory comparison among SLAM results and Ground truth

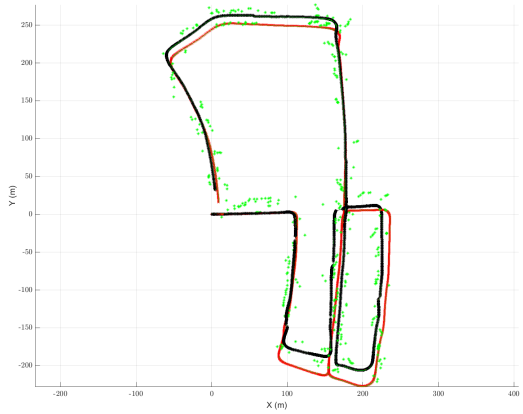


Fig. 6. Soft association locolization result, with landmark marked as green stars.

TABLE II
SE(3) ERROR ON KITTI SEQUENCE 18

| Method | SE(3). error [1/m] |
|---|---|
| with hard association | 31.3158 |
| with soft association | 18.0414 |

## V. CONCLUSION

*1) Visual Odometry and Landmark Detection:* The results of estimated trajectory provided by the monocular visual odometry were decent as seen in the error analysis above. These results were improved by performing the algorithm on increments of 100 camera frames. As seen in Figure 2 the monocular visual odometry results did not lead to the poses overlapping where loop closure should have occurred. This could have potentially been an issue which prevented the landmark measurements from being able to improve the SLAM result.

For the object detection, our result(see in our demo video for Mask RCNN) is not always accurate. There are some mis-detection here, some objects are detected as vehicles even if they are not actually vehicles, and this only happens for some frames. Thus the issue is that from the demo video we can see some noise, which means some vehicles only shows up for several frames. This is not what we expected, however, this issue is tolerable to our SLAM algorithm. The big issue is about the accuracy of our landmark location.

The 3D location of landmarks are relatively accurate only for those that are within a small range (about $60m$) from the Lidar. So we can only use these relatively accurate landmark locations. However, these landmark locations are just roughly accurate, this produces a big inaccuracy for the global location based on the visual odometry, which generates drifts in the localization part.

*2) iSAM2 Graph Optimization:* Both the results from iSAM 2 with hard and the soft data associations are not better than the trajectory estimate provided by the monocular visual odometry, according to the error calculated. One possible reason for this is that loop closure was not achieved and thus the best performance for the algorithm would come from the information provided for the relative pose transformations. The main reason for failing to loop closure can be summarize into two. For the first reason, the measurements we get is noisy and unreliable. The moving cars are not being removed from the observation. The bounding box may appear when there is not a full car shown, even in some extreme case, the MASK RCNN will false detect the shadow of a car to be a real car. The second reason is that we still using the tradition way for loop closure, that is only the geometric information being considered. As shown in the result, even just with the visual odometry, the accumulated error at the point which loop closure should happen is 10m. Some state of the art loop closure method like using the HOG features histogram to set up the pose constraint could be preferable choice.

*3) Overall:* Based on the error analysis seen in the preceding results section, the overall results for the trajectory estimate are improved by performing the soft associations and

the theory and experiments introduced in [2] are validated. By including the data associations in the factor graph, the optimization process has more degrees of freedom which allows the solver to approach a more robust solution.

## APPENDIX

### A. Weight calculation for soft data association

**Algorithm 1** SS_weight

L ← nnsearch($x$,$z$)
//calculate nearest neighbour first
**for all** measurement $z$ **do**
    $M(z_i)$ ← Maha_norm($L$,$z_i$)
    $w(z_i)$ ← 1-$M(z_i)$/sum($M(z_i)$)
    $w(z_i)$ ← normalize($M(z_i)$)
    //the weight should sum up to 1
    $w$ ← [$w$  $w(z_i)$]

### B. Measurement factor update for soft data association

**Algorithm 2** measurement_update

**for all** measurement $z$ **do**
    **if** $L_i$ == 0 **then**
        initialize_new_landmark($x_t$,$z_i$)
    **else**
        **for all** landmark in $L$ **do**
            new_range_factor($L_j$,$z_i$,observation_noise/$w_{ij}$)


## ACKNOWLEDGMENT

Thank you to our professor, GSI's and graders. This class has been a great experience and we are looking forward to applying what we have learned to solve challenging engineering problems.


## REFERENCES


[1] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (slam): part ii. *IEEE Robotics Automation Magazine*, 13(3):108–117, 2006.

[2] Sean L. Bowman, Nikolay Atanasov, Kostas Daniilidis, and George J. Pappas. Probabilistic data association for semantic slam. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1722–1729, 2017.

[3] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *I. J. Robotic Res.*, 25:1181–1203, 12 2006.

[4] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, June 2006.

[5] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[6] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[7] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2:31–43, 2010.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, Oct 2017.

[9] M. Kaess, A. Ranganathan, and F. Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, Dec 2008.

[10] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[11] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.