Dynamic Semantic Mapping by Scene Flow Propagation

Lu Gan[1], Yiqian Gan[1], Ganesh Sevagamoorthy[1], Abhishek Venkataraman[1]

***Abstract*— This project tackles the problem of semantic mapping in dynamic environment using a Bayesian filtering scheme. Instead of only containing metric information, our 3D map has both occupancy and semantics belief which allows scene understanding, object detection and tracking. We inferred those information incrementally from sequential stereo images by using scene flow to do propagation, image depth and segmentation to do correction. We give the implementation details, experimental results and how we used our knowledge from mobile robotics to analyze the results in the report.**

I. INTRODUCTION

For autonomous vehicle, Simultaneous Localization and Mapping (SLAM) in dynamic urban street scenes with an appropriate scene understanding ability is essential to maintain road safety for both drivers and pedestrians and to ease the further motion planning problem. It not only requires persistent and accurate 3D model of static surroundings, but the same accuracy and persistence for the dynamics environment. It should be able to parse and update both position and semantic information for static as well as dynamics objects.

In this project, we implemented a method to filter geometric, semantics and motion cues in a consistent map [1]. The image-based semantic and occupancy beliefs in each voxel are propagated and updated in the dynamics scenes. The propagation uses scene flow information and the camera pose. The update step incorporates the observations namely, the depth and probabilistic distribution for the labels to develop a dynamic consistent map.

In the experiments, we first demonstrated our semantic octree implementation by visualizing the 3D voxel grid built from Stanford 2D-3D-Semantics Dataset [2]. Our algorithm is then evaluated on the KITTI benchmark suite [3]. To extract the semantic and occupancy information, we use methods in [4] to pre-process stereo images. Then, information is discretized and stored into voxels. After that, particle filter is used to propagate and update the 3D voxel map. Further details about the methodology and dataset we used in this project are given in the following sections.

Our implementation is available in github[1].

[1] ganlu,ganyq,gse,abhven@umich.edu

[1]https://github.com/ganlumomo/DynamicSemanticMapping

Fig. 1: Occupancy semantic map. Top: Image from the KITTI scene flow dataset. Bottom: Color-coded local measurement map using our method.

II. RELATED WORK

SLAM is one of the most important ingredients for autonomous driving. Also, it is hard to achieve high quality consistent 3-D mapping in the dynamics street scenes. Current state-of-art methods such as ORB-SLAM [5] or LSD-SLAM [6], demonstrate high performance for large-scale trajectory estimation and 3D reconstruction. But they are limited to scenes with a few moving objects since these methods are derived inherently from the method designed for static map. Other methods such as SLAMMOT [7] and KinectFusion [8] have been proposed to distinguish static environment and moving objects . These methods could exclude dynamic objects to achieve higher quality of tracking and mapping. However, they cannot incorporates scene semantics such as categorization of point clouds into labels of cars, pedestrians, roads or buildings, which could make huge difference in the motion planning stage.

Works where semantic segmentation is used could be distinguished by how semantic segmentation are parsed and fused into global map. Koppula et al. [9] proposed a semantic mapping using RGBD sensors. They applied the Markov random field model to the labeled point clouds with appearance and geometric features. Stuckler et al. [10] proposed a semantic SLAM approach which filter semantic segmentation using RF classifier in multi-resolution surfel maps. To achieve semantic SLAM, Sengupta et al. [11] proposed a method to segmented stereo images into object class with conditional random field (CRF) [12] and fuse segmentations into ground plane projection. However, it assumes the scene to

remain static during mapping.

III. Overview

In this project, our main goal was to investigate some of the techniques primarily described in [1]. The fundamental idea is to fuse disparity map and semantic segmentation of 2D sequential stereo images into a persistent 3D voxel grid. In order to build a spatially as well as temporally consistent map, dynamic object movements in the environment should also be taken into consideration. Paper [1] formulated this problem into a voxel grid map state estimation problem, and solved it by using recursive Bayesian filtering algorithm.

This recursive algorithm can be broken down into two major steps, i.e, prediction and correction. In the prediction step, we propagated all the voxels with scene flow information by sampling from a Gaussian distribution centered at the scene flow and with some uncertainty. After this, we integrated the current 2D observation into a local 3D representation and used this as the likelihood to correct the predicted belief in the correction step.

As what we wanted to do is a large-scale three dimensional mapping, which needs a lot of computational time and memory, we initially spent some time to try different implementing methods. The original paper used voxel hashing [13] to deal with this problem, which is an efficient CUDA implementation for voxels, proposed to be used in surface reconstruction represented by Truncated Signed Distance Field (TSDF) instead of occupancy map. However, considering the time constraint for our project and the complexity of GPU programing, we ended up using an open source C++ library OctoMap [14] to implement our voxel class and functions. OctoMap uses octree to represent the full 3D model where every node except the leaf nodes has 8 children. By using some of the embedded voxel-based operators, such as probability-log odds conversion, point counting, we are able to save our time and energy to work on the most important part of this project.

IV. Methodology

A. Data Preparation

1) Semantic Segmentation: Semantic segmentation has been an area of key interest in perception and more specifically in computer vision. It is the task of grouping the pixel in an image and classifying them into categories. It plays an important role in understanding the overall structure of the environment of the autonomous vehicle. Figure 2 illustrates the semantic segmentation of an image into 12 categories using SegNet. Traditionally, most of semantics segmentation is done on images, i.e. on pixels, which are a array of dimension 2. In this project we refer to them as 2D semantics. The same segmentation can be extended to 3D space in terms of voxels. We will be referring to this as 3D semantics.

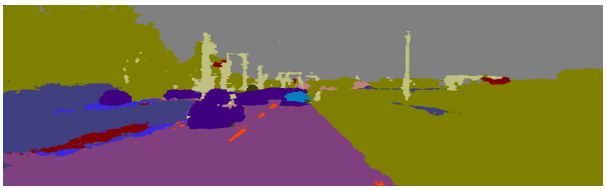

Fig. 2: Semantic segmentation of road image into 12 categories using SegNet.

2) Scene reconstruction from stereo images: Since we are interested in updating the information of the environment in 3D space, the first step is to get 3D points from the stereo image pair. At each time step, the disparity map is computed, which is the disparity of features between left and right image measured in pixels. The world points in the camera frame of reference can be computed using Equations 1 - 3

$$z_{world} = f \times \frac{b}{d_{(x,y)}} \tag{1}$$

$$x_{world} = (x_{image} - c_x) \times \frac{b}{d_{(x,y)}} \tag{2}$$

$$y_{world} = (y_{image} - c_y) \times \frac{b}{d_{(x,y)}} \tag{3}$$

where, c_x and c_y are the principle point of the camera, b and f are the baseline and the focal length respectively[2]. x_{image}, y_{image} are the observed points on camera frame.

In order to reduce the memory occupied by voxels, we did not store any points which were more than 50 meters away.

3) Scene flow: In images, optic flow is used to map the correspondence between two subsequent images by computing a flow 2D flow vector (u,v) for every pixel in the initial frame. Menze et. al[4] created a novel method to compute scene flow, which a 3 dimensional flow in the x,y,z axis. In our project, we used the publicly available code `https://github.com/yuvval/summer_vision_lang_2/tree/master/objectsceneflow` to get the scene flow.

When applied to two pairs of sequential stereo images, this method outputs two disparity maps (one for each time step) and true optical flow. In order to compute the 3D scene flow, we compute δz using the two disparities (d_0, d_1) using equation 5. We then compute δx and δy using equations 6,7.

[2]all the camera parameters c_x, c_y, b, f are obtained from the camera calibration file provided in KITTI dataset.

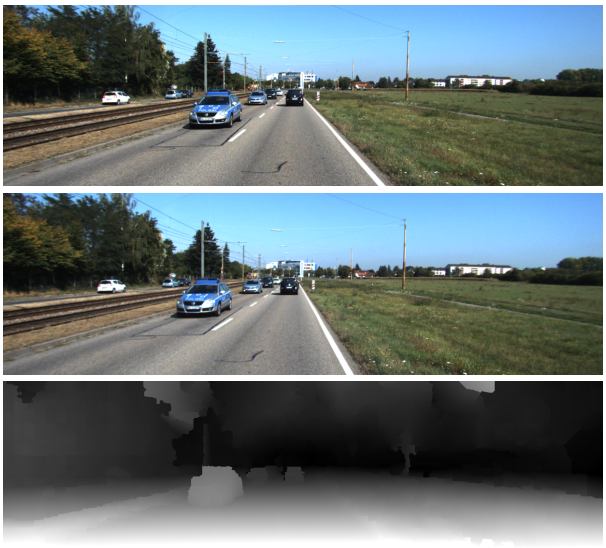

Fig. 3: Top: Left stereo image Middle: Right stereo image Bottom: Disparity Map for the pair (disparity map enhanced for visibility)

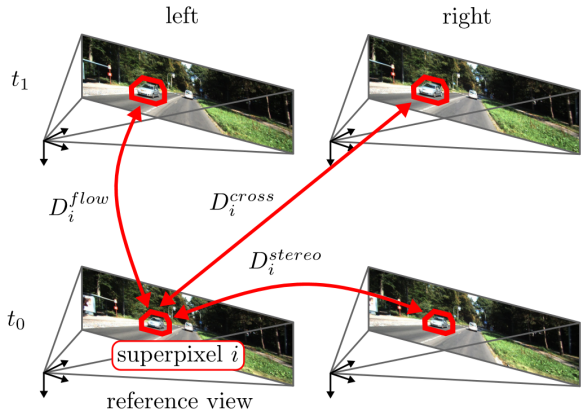

Fig. 4: Images used for computation of scene flow [4]

$$z_1 = z_0 \times \frac{d_0}{d_1} \tag{4}$$

$$\delta z = z_1 - z_0 \tag{5}$$

$$\delta y = (y_{image} - c_y + v_{(x,y)}) \times \frac{b}{d_1} - (y_{image} - c_y) \times \frac{b}{d_0} \tag{6}$$

$$\delta x = (x_{image} - c_x + u_{(x,y)}) \times \frac{b}{d_1} - (x_{image} - c_x) \times \frac{b}{d_0} \tag{7}$$

where (u,v) are the true optical flow vectors in the image plane in x and y directions respectively.

Figure 5 plots the optic flow as image. The magnitude of the direction vectors (u,v) are represented by the color of the image.

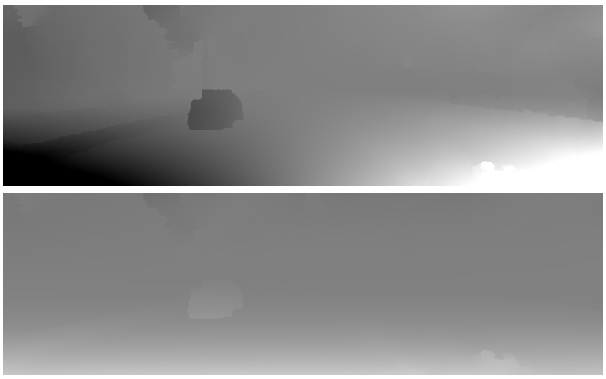

Fig. 5: Top: magnitude of u, true flow in x. Bottom: magnitude of v, true flow in y. Computed using method in [4]

4) Visual Odometry: In mobile robots, camera is generally mounted on the robot. Thus, making the observation frame different from the world frame. The method mentioned above, compute the position and flow with respect to this moving frame. In order to account for the ego-motion of camera, we compute the camera pose using visual odometry. Visual odometry return a 4x4 transformation matrix, which can be used to convert the homogeneous 3D points from the car frame to world reference frame.

B. Prediction

In real environments, autonomous vehicles would encounter dynamic objects as well. In order to keep track of these, we need to ensure that the position of dynamic objects are updated with time. In our project, we account for this change by propagating the points in previous frame with the scene flow for current frame. The scene as mention in the previous section, is in the frame of reference of the camera. Hence, in order to get the true motion, we compensate the scene flow with the ego motion of camera using visual odometry.

In prediction step, we determine the probability of semantic and occupancy belief of last time step according to scene flow estimate of the current time step. As described in [1] the probabilities are given by Equation 8. Semantic belief and occupancy belief are considered to be independent and can be calculated independently using the Bayesian filter.

$$p\left(y_{t,j} \,|\, X^{t-1}, F^t\right) = \sum_y \sum_k p\left(y_{t,j} \,|\, y_{t-1,j} = y, f_{t,k}\right) p\left(y_{t-1,j} \,|\, X^{t-1}, F^{t-1}\right) \tag{8}$$

where, F_t is the scene flow estimates up to time t and X_t is the observations till time t. We propagate some

uncertainty in order to account for the error in scene flow measurement. Hence, we determine an intermediate distribution $\tilde{y}_{t,j}$ using equations,

$$p\left(y_{t,j} \,|\, X^{t-1}, F^t\right) = \sum_y p\left(y_{t,j} \,|\, \tilde{y}_{t,j} = y\right) \\ p\left(\tilde{y}_{t,j} = y \,|\, X^{t-1}, F^t\right) \tag{9}$$

The prediction step comprises of two parts. First, we use particle filter to propagate the belief of the previous state according to current scene flow estimate Equation 10. Weight of particles is equal to the inverse of number of particles in its originating voxel. The number of points in every voxel is not equal; voxels closer to camera having more points than the ones further away. Hence, instead of having equal number of particles in each voxel, the number of particles was determined by the number of points in each voxel. Thus more particles are selected from voxels near to the camera than the ones further away. Next, we sample the points in the point cloud with the scene flow estimate to propagate them. The scene flow distribution is assumed to be a Gaussian distribution with mean equal to scene flow and error equal to $\sum_f = 2\sum_{u,t}$ where, $\sum_{u,t}$ is the covariance associated with error in stereo depth estimate. The occupancy and semantic belief of these points are stored in a temporary tree. Once all the points with scene flow are propagated, the originating voxels in the original tree are deleted.

$$p\left(\tilde{y}_{t,j} = y \,|\, X^{t-1}, F^t\right) \\ \propto \sum_{i \in P_k^i(f_{t,k})} w_{t-1,k}^{[i]} p\left(y_{t-1,k} = y \,|\, X^{t-1}, F^{t-1}\right) \tag{10}$$

where, $P_k^i(f_{t,k})$ are particles originating in voxel k that end up in voxel i.

The second step in prediction is smoothening the semantic and occupancy belief. All voxels in the temporary tree are smoothened using Equation 11. These voxels are then normalized and updated to the original tree. This updated original tree is then used for correction.

$$p\left(y_{t,k} = y \,|\, \tilde{y}_{t,k} = y\prime\right) = \begin{cases} \delta, & \text{if } y = y\prime \\ \frac{1-\delta}{N-1}, & \text{if } y \neq y\prime \end{cases} \tag{11}$$

where, δ is the smoothening factor and N is the number of state variables.

C. Correction

Once we finished scene flow propagation, corresponding to predicting the state according to motion model in Bayesian filter framework, we correct the error introduced by the uncertainty of scene flow by integrating our sensor model. Here, our sensor model is the stereo depth for occupancy belief and 2D semantic segmentation for semantics belief, respectively. Essentially, our sensor model gives us the 2D observations of true current occupancy and semantic state of the real 3D environment. If observations accord with our prediction, the likelihood of our prediction should be high; Otherwise, it should be low to correct our prediction. Two steps are exploited to achieve correction and the technical details are given in this section.

1) Local Measurement Map Construction: First, we need to transform the 2D observations into the sensor model that we can directly use in 3D mapping. Thus, a local measurement 3D map is built purely from observations by fusing pixel depth and semantics information into voxels. More specifically, the objective of this process is, given a frame of pixels with the corresponding world coordinates x, y, z as well as their semantic labels l, determining the occupancy and semantics belief for all the voxels involved.

The way we determine occupancy and semantics belief are different. To calculate the occupancy for a specific voxel, we first need to determine the voxel as measured occupied or free. The original paper achieved this by projecting a specific voxel into the the stereo frame and determining if it lies in front of the measurements (the depth), which we thought was very inefficient. Instead of projecting voxels, we calculated the world coordinates of each pixel as the endpoint of a ray starting from the left camera, and used 3D Bresenham's line algorithm[15] to discretize the ray into voxels to obtain the correspondences between 2D pixels and 3D voxels. For each ray, the voxel in which the endpoint is placed is considered as occupied, and all the voxels along the ray between camera and endpoint are considered as free. We added a different constant occupancy logodds to occupied voxel and free voxel separately. And after we processed all pixels in this way, the occupancy belief of all voxels in the local measurement map is obtained. The methodology we use to determine semantics belief is different from occupancy, because we cannot infer geometric relationship from semantic observation. Thus, we accumulate the 2D semantic segmentation into local measurement map by averaging the label distribution of pixels that fall into a voxel:

$$p\left(l_{j,t}^m \,|\, x_t\right) = \frac{1}{|U(j)|} \sum_{u \in U(j)} p\left(l_u \,|\, x_t\right) \tag{12}$$

where $U(j)$ is the set of pixels within the voxel j, and $|U(j)|$ is the number of elements in this set.

2) Global Voxel Map Update: After we finished the local measurement map construction, we use it as our sensor model by integrating it into the global voxel map

using Bayesian updates:

$$p\left(y_{t,j} \mid X^t, F^t\right) = \eta \frac{p\left(y^m_{t,j} \mid x_t\right)}{p\left(y_{t,j}\right)} p\left(y_{t,j} \mid X^{t-1}, F^t\right) \quad (13)$$

where $p\left(y^m_{t,j} \mid x_t\right)$ is the posterior probability of y given observation x at time step t for voxel j, which is represented as belief in voxel j in the local measurement map, $p\left(y_{t,j} \mid X^{t-1}, F^t\right)$ is the belief after prediction in voxel j in the global map, corresponding to $\overline{bel}(y)$ in our text book [16], $p\left(y_{t,j} \mid X^t, F^t\right)$ is the belief after correction, corresponding to $bel(y)$, $p\left(y_{t,j}\right)$ is uniform priors and η is a normalization factor. By using Bayesian equation, we are able to update the occupancy and semantics in the current state.

V. EXPERIMENTS & RESULTS

A. Semantic Voxel Grid

We implemented our semantic voxel grid based on the OctoMap C++ library by inheriting from a basic class OcTree and OcTreeNode. This library provides data structures (octree) and some occupancy operators we can directly utilize. We added the color and semantic information to our SemanticOcTreeNode class, and implemented all the functions to operate color and semantics by ourselves, as well as all the prediction and correction steps.

The first thing we want to test is voxel grid construction, because it is the fundamental part of the whole implementation. To simplify this experiment and be able to check the results more directly, we chose Stanford 2D-3D-Semantics Dataset (2D-3D-S) [17]. This dataset offers semantically annotated aligned point cloud, which is very suitable to our test. But we also realized that we were not able to use this dataset to validate our whole project, because it does not offer sequential images with time stamps which is the key to building a temporally consistent map. Some of the 3D voxel grids we built to verify our implementation are shown in Fig. 6 and 7. We used the viewer Octovis provided by OctoMap to visualize our semantic octree.

Fig. 6 is an occupancy color grid without showing semantic information, which contains occupancy belief and color information in each voxel. The color information of a voxel is calculated by averaging the RGB value of all the pixels that fall into that voxel. Fig 7 shows an occupancy semantic grid for a conference room with different objects, e.g., chairs, table, wall, celling, and floor, where the semantic labels are color coded. Here, each voxel additionally contains a semantic label, and the label for a voxel is obtained by averaging the label distribution of all the pixels, as the same way we fused the color information.

Fig. 6: Occupancy color grid built from Stanford 2D-3D-Semantics Dataset. RGB information is fused from pixels into voxels.

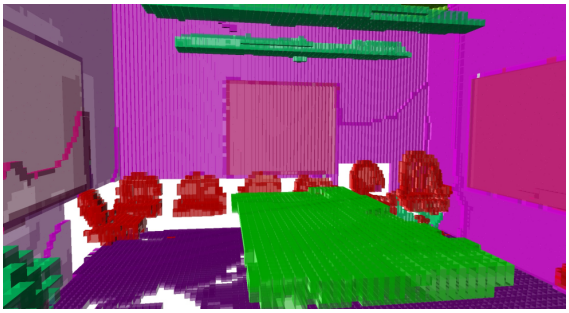

Fig. 7: Occupancy semantic grid built from Stanford 2D-3D-Semantics Dataset. Semantic labels are color coded.

From the visualized results, we verified our implementation. And one need to notice that the way we built the occupancy semantic tree here is actually the same procedure of building a local measurement map in the correction step, which has been explained in detail in Section IV C. Essentially, this is the way how we interpret and integrate 2D observation into 3D voxel grid in this project. Therefore, the results also demonstrate the correctness of our local measurement map construction.

B. Prediction

Results for the prediction step is shown in Fig. 8. The middle and right figures are maps obtained after prediction without and with uncertainty, respectively. It can be noticed that three cars in the road moved according to scene flow, whereas there was no change to the environment. It can be seen in the result how different uncertainty of the scene flow measurement affects the propagation.

It is seen that with increase in covariance of noise random variable, the particles after sampling become more spread out changing the belief of more voxels, thus the map becomes denser. We used a constant value

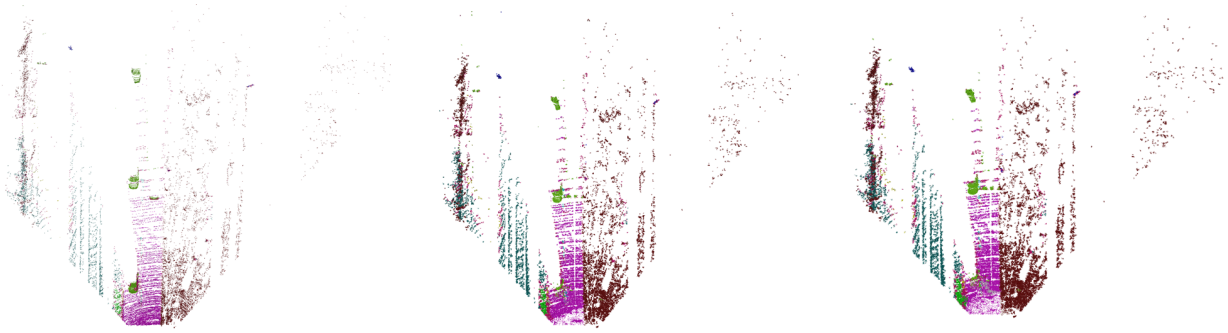

Fig. 8: Prediction Result. Left: Global voxel map before prediction. Middle: Global voxel map after prediction without uncertainty. Right: Global voxel map after prediction with uncertainty.

of covariance matrix in our implement, as we did not have an accurate estimate of the correct covariance due to camera-based measurement method. In experiments, we also found out that this covariance matrix is actually much more important than what we thought before. It has direct influence on the success of particle filter and other sampling-based filters. When we tried to set a large covariance to the noise, the distribution of semantic label was destroyed and could not be corrected. Thus, we chose a relative small covariance value in our final experiment. Further it can be seen from Fig. 8 that for voxels closer to the camera, the error in sceneflow is much higher.

Another important observation was the use of Particle filter instead of Kalman filter. This was because, there was no explicit motion model for scene flow. Particle filter only applied motion model, thus, did not need a closed form motion model.

Special care had to be taken to get rid of dynamic particles moving out of the frame between consecutive frames. This was important as it avoids storing information about dynamic objects in our map which are not within the our perceptual range.

It was also noted that the paper [1] didn't account for the difference between two destination voxels which receive different number of particles. It assumes that a destination voxel receiving 1 particle and another voxel receiving 100 particles, each with same semantic belief and occupancy belief as equal.

C. Correction

We used the same sequential images to test our correction steps. The prediction step propagated voxels in previous one time step based on scene flow, which gave a prediction of map state in current time step. Then, the correction steps try to update this prediction based on observations in current time step. The results of both local measurement map construction and global voxel map update are given below.

1) Local measurement map: The local measurement map is built purely from 2D observation of successive images by updating occupancy log odds and averaging semantic distribution. It does not have any previous information. The local measurement map is built from the image in time step t is given in Fig. 9.

2) Global voxel map: After the local measurement map has been successfully constructed, we used this map as our likelihood in Bayesian filter to correct the belief in every voxel in the global map, according to Equation 13. The occupancy belief and semantics belief are updated separately and independently. Our correction results are given in Figure 10.

Because of time constraint, we were not able to come up with a whole sequence of reasonable input data, which is also one of the drawbacks of this method, we ended up doing an experiment only on a few successive images, among which there is no dramatic camera motions. That is why the global voxel map here does not look larger than the local map. However, since our Bayesian updating algorithm is recursive, we think the result is enough to demonstrate the effectiveness of our method.

From Figure 9 we can notice that there is an big error in 2D image segmentation, where the point cloud of the car in the bottom of the figure is classified into two different classes (represented by green color and red color). The error is brought by the uncertainty of our observation. In addition, the uncertainty of our motion model (scene flow) also gives error in the prediction. For the same car in Figure 10, we can barely tell the shape of the car there because the corresponding voxels were propagated to somewhere else because of scene flow error.

Every probabilistic motion model and measurement model has uncertainty which brings error to prediction and estimation. Bayesian filter reduces uncertainty of the estimate of current state by integrating both uncertainties of motion model and sensor model, which is clearly

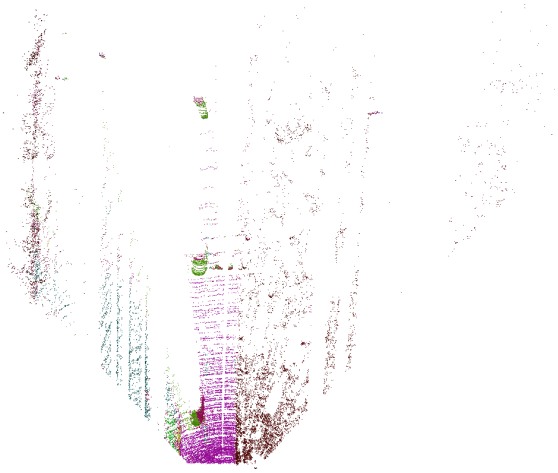

Fig. 9: Local Measurement map built from the current observation.

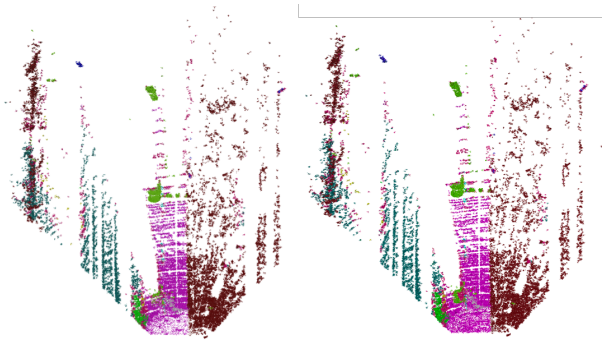

Fig. 10: Correction Result. Left: Global voxel map after prediction. Right: Global voxel map after correction.

illustrated in Fig. 10. The voxels corresponding to the car showed up again without incorrect semantic labels (represented by red color) in the global voxel map after correction, which successfully demonstrated the effectiveness of our Bayesian correction, i.e., reducing both prediction and observation error.

VI. CONCLUSIONS

In this project, we have implemented a method for 3D semantic mapping in the dynamics scenes. We incorporated scene flow to propagate occupancy and semantic and ensure a temporal consistent map for both static and dynamic objects. Then Bayesian filter is used to update the dynamics objects so that semantic map is consistent globally.

During this project, we realized that the original paper only used a heuristic way to do propagation, and did not provide a probabilistically sound framework, which we want to improve in the future. We also noticed that the uncertainty of prediction model is very critical to the success of particle filter. Trying to get a better estimate of the covariance will also improve this method.

Further extensional work could be done for object-oriented semantic SLAM. For example, object detections could be done after constructing a global semantic map. Further motion planning problem could also be studied such as trajectories prediction for the dynamics objects.

ACKNOWLEDGMENT

We would like to thank Dr. Gonzalo Ferrer, Mr. Sudhanva Sreesha and our fellow students in EECS 568 for the discussing and inspiring us through out the duration of the project. We are also thankful to Mr. Arash Ushani and Mr. Vikas Dhiman for their inputs in understanding the dataset.

REFERENCES

[1] D. Kochanov, A. Osep, J. Stückler, and B. Leibe, "Scene flow propagation for semantic mapping and object discovery in dynamic street scenes," in *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016, to appear.

[2] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese, "Joint 2D-3D-Semantic Data for Indoor Scene Understanding," *ArXiv e-prints*, Feb. 2017.

[3] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[4] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[5] M. J. M. M. Mur-Artal, Raúl and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[6] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*. Springer, 2014, pp. 834–849.

[7] C.-C. Wang, C. Thorpe, and S. Thrun, "Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 1. IEEE, 2003, pp. 842–849.

[8] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, 2011, pp. 127–136.

[9] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena, "Semantic labeling of 3d point clouds for indoor scenes," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, ser. NIPS'11. USA: Curran Associates Inc., 2011, pp. 244–252. [Online]. Available: http://dl.acm.org/citation.cfm?id=2986459.2986487

[10] J. Stückler, B. Waldvogel, H. Schulz, and S. Behnke, "Dense real-time mapping of object-class semantics from rgb-d video," *J. Real-Time Image Process.*, vol. 10, no. 4, pp. 599–609, Dec. 2015. [Online]. Available: http://dx.doi.org/10.1007/s11554-013-0379-5

[11] S. Sengupta, P. Sturgess, P. H. Torr *et al.*, "Automatic dense visual semantic mapping from street-level imagery," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 857–862.

[12] V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," *Adv. Neural Inf. Process. Syst*, vol. 2, no. 3, p. 4, 2011.

[13] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 169, 2013.
[14] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com
[15] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems journal*, vol. 4, no. 1, pp. 25–30, 1965.
[16] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.
[17] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3d semantic parsing of large-scale indoor spaces," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.