2019 Third IEEE International Conference on Robotic Computing (IRC)

A Review of SLAM Techniques and Security in Autonomous Driving

Ashutosh Singandhupe, Hung Manh La

***Abstract*—Simultaneous localization and mapping (SLAM) is a widely researched topic in the field of robotics, augmented/virtual reality and more dominantly in self-driving cars. SLAM is a technique of building a map of the environment and estimating the state of the robot in the map in which it is moving, simultaneously. SLAM has been there for more than 30 years and has contributed significantly in the industry targeting from small scale driven applications to large scale, which resulted in the advent of this decade's self driving cars. This paper attempts to give an understanding and progress of SLAM in autonomous driving industry as well as briefly describes the SLAM techniques that have contributed significantly to the industry, which were especially evaluated on KITTI dataset. We have also attempted to compare various techniques that were presented and made a rough estimate on why the state of the art approach can be revised and refurnished to suit the complex understanding of the environment for effective localization. In the end we have briefly described the security threats related to autonomous driving industry and why this is alarming.**

I. Introduction

The ease with which most animals and we human beings navigate in the environment, perhaps, can be replicated in robots as well. This complex process of navigation, no wonder how well we do, can not be easily represented mathematically. The only way that dumb robots can be made to navigate in an environment is to represent the environment in some simpler forms, which can be algorithmically justified. Simultaneous Localization and Mapping (SLAM) is an algorithmic process of a robot/sensor system, which involves perceiving the environment using sensors and estimating the position of itself in the environment simultaneously [7]. For a robot, the environment can be represented as a culmination of different geometrical structures (landmarks, obstacles etc.) also called as map, and the pose can be represented as position and orientation of the robot, and it is generally termed as robot state. The map assists the human operator in visualizing an unknown environment and setting up the path of the robot for navigation. Another significant advantage the map provides is that it helps in minimizing the error while estimating the robot state (pose) during navigation. Keeping track of visited landmarks the robot can detect a loop closure in case it has revisited a location, thereby minimizing localization error, which is quite a similar to as how we human beings navigate. Automatic navigation is also significant part of robot navigation research, which has resulted in numerous algorithms such as Rapidly exploring Random Trees (RRT), extended RRT (RRT*), Rapidly-exploring Random Graph (RRG), Probabilistic Roadmap (PRM), etc., [9]. These algorithms have directed many researchers to explore and improve robot navigation in complex environments. However, the current state of the art demands for a more improvement since it is yet to be fully solved for real time dynamic environments.

SLAM is a heavy component in autonomous driving car systems. This paper attempts to explore the various techniques tested on autonomous driving cars with reference to KITTI dataset [1] as our benchmark. The drive for SLAM research was ignited with the inception of robot navigation in Global Positioning Systems (GPS) denied environments. Although GPS improves localization, numerous SLAM techniques are targeted for localization with no GPS in the system. Initially, probabilistic estimation techniques were introduced like Kalman Filters (KF), which were later extended to Extended Kalman Filters (EKF), and Unscented Kalman Filters (UKF) for non-linear systems [7]. Particle filters like Rao-Blackwellized and Monte Carlo filters have also contributed significantly to the SLAM research [7]. Another approach that has grabbed attention is the graph based SLAM where the robot pose is represented as a node/vertex in a graph, and the edges represent the errors in measurements from various sensors. Subsequently the process involves generating a pose graph and minimizing the error using mathematical techniques like Gauss-Newton/LevenbergMarquardt [18]. SLAM techniques like Oriented fast and Rotated Briefs-SLAM (ORB2-SLAM) [24], [25] are the graph-based localization. Another interesting approach grabbed attention since the advent of deep learning with focus on Convolutional Neural Networks (CNN). Quite interesting results were observed especially with the work on CNN-SLAM [37]. It was shown through experiments that robot pose or localization could be achieved from a pair of images acquired by a moving robot through deep learning or CNN. Even though the CNN-SLAM approach is promising, this approach has invited few challenges that needs to be addressed. Deep learning requires high-end Graphic Processing Unit (GPU) systems, which is still a challenge for robotic embedded systems. Moreover SLAM systems are seen to be directed on continuous open-world scenes where the environment keeps changing. These changes needs to be learned on a continuous basis for a deep learning system. To our best knowledge, deep learning has not significantly evolved in the current state of SLAM to learn the dynamic changes in the environment robustly. From the various techniques introduced in SLAM, one can observe that SLAM has an inclination to combine various fields like signal processing, deep learning (CNN-SLAM) and more significantly computer vision.

The whole approach of SLAM is based on a robot localizing itself given the sensor measurements. However the traditional SLAM algorithms do not extend to perform task of a robot driving itself to collect more information about the environment. In a more general sense, traditional SLAM algorithms only estimates it's localization, when it is navigated

Ashutosh Singandhupe and Dr. Hung La are with the Advanced Robotics and Automation (ARA) Laboratory, Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557, USA. Corresponding author: Hung La, email: hla@unr.edu

This material is based upon work supported by the National Aeronautics and Space Administration (NASA) Grant No. NNX15AI02H issued through the NVSGC-RI program under sub-award No. 19-21, the RID program under sub-award No. 19-29, and the NVSGC-CD program under sub-award No. 18-54.

978-1-5386-9245-5/19/$31.00 ©2019 IEEE
DOI 10.1109/IRC.2019.00122

or assisted by external source or through an external command. There is no 'conscious' effort from the robot to navigate itself and collect the data from the environment. Active SLAM, exploration and localization done at the same time, is an approach that attempts to solve this problem. Active SLAM basically tries to solve it in 3 steps. The first step specifies the robot trying to find possible actions (e.g., turn right, turn left, forward, backward, path selection, etc.) that it could take given the map space, however it exposes the challenge of increased computational complexity. The second step says that, even if an action is confirmed to be taken, the logic behind performing that action needs to be justified with respect to the goal of the task, as well as the complexity of the future action that could be taken to achieve that goal. The final step indicates that even if the action is performed it is quite difficult to arrive to a conclusion whether the exploration task has been completed or not. Based on our knowledge, active SLAM still requires mathematical proofs at various aspects [7].

Semantic SLAM is another active area of research for solving the SLAM problem. Based on this approach, the information of the environment is stored or perceived as semantics, where each part is seen as distinguished geometrical object. In more simpler terms, semantic SLAM tries to build the map of the environment using objects as references. In traditional SLAM methods, the whole approach of map building was done using the idea of aligning points and planes. However, the semantic SLAM approach uses the idea of aligning point clouds using objects, which are referred or understood as semantics. This has pushed SLAM community to create more techniques, which are tasks based rather than path plannings based. It will also create a room for effective robot-human interaction. The way it differs from previous SLAM techniques is that previous approaches were based on already seen environments (regardless of identification of different objects) while semantic SLAM works on identifying a place based on objects described as semantic labels. However, it presents a challenge, which involves fusing semantic information from various sources at different times. The problem of consistently fusing it, still persists [7], [8].

Despite the tremendous progress made in SLAM in the past 30 years, one question still bothers the robotics community, Is SLAM solved? [7]. Our understanding is that SLAM is an estimation problem. Given the complexities of the environment and the uncertainties in the sensor measurements, it is yet to arrive a complete solution. As of our knowledge, SLAM is still unsolved. A good solution significantly relies on the environment, the robot, the uncertainties in the sensor measurements and the level of performance that we intended to achieve.

This paper primarily focuses on reviewing various SLAM techniques that were tried and tested on autonomous driving cars based on the KITTI dataset [1]. We are risking an attempt to classify various SLAM techniques, which are based on sensors used for localization and the ability of the SLAM algorithms to detect a loop closure. A loop closure is a technique of detecting a visited landmark or a scene in an environment. As of our knowledge, very few of the state-of the art algorithms we have encountered has solved the loop closure problem with respect to this autonomous driving car dataset.

The remainder of the paper is organized as follows. Section II introduces techniques, which is further divided into Light Detection and Ranging (Lidar) based techniques (including Lidar and monocular camera) and stereo-based SLAM techniques. Later on, in section III we also intend to explore the security aspects in relation to autonomous driving and explore briefly the threats associated with it. Finally, conclusions are discussed in Section IV.

II. SLAM IN AUTONOMOUS DRIVING

Since autonomous cars require localization to navigate in the environment, it is quite obvious to explore the SLAM techniques in relation to autonomous cars. Most autonomous cars use Lidars and stereo cameras to perceive the environment in which it is navigating. However, most techniques that we have encountered so far, either have attempted to solve the localization problem on Lidar and Monocular camera or Stereo camera.

A. Lidar Based Odometry

There are numerous algorithms written for estimating odometry using Lidar. Implicit Moving Least Squares SLAM (IMLS-SLAM) [12] is quite popular, which uses scan-to-model matching framework. Initially, it uses an algorithm to sample the 3D scans and uses IMLS for surface reconstruction, which is claimed to have an improved matching quality. One key factor to note in this work is that it uses only 3D Lidar sensors for odometry estimation. The work claims to perform better results than the state of the art algorithm for odometry estimation, Lidar Odmetery and Mapping (LOAM) [43], [44]. However, the KITTI website [15] shows that LOAM outperforms every other algorithm that has been tested on all the odometry datasets. This contradiction needs to be evaluated further.

Another work by [17], also called as Lidar-Monocular Visual Odometry (LIMO), have proposed a method, which fuses data from both Lidar and monocular camera. It first calculates the camera features and estimates the depth using the Lidar data corresponding to those features. Fusing the data together, it estimates the motion using a technique called bundle adjustment. The system for this algorithm can be better explained in steps/blocks. The first block relates to the camera, where it extracts the features. It includes feature tracking step and feature association step. Feature tracking is done using the Viso2 library. The feature association step is mostly related to extracting the depth of the camera features using the highly precise Lidar data. To perform these, the Lidar point cloud is transformed into camera frame and projected into the image plane. Then, for each features the following steps are performed. (1) From a projected set of Lidar points, the algorithm choose a set of points that are around a given feature. They use a rectangle to define the neighbourhood around that point. (2) Then it performs a plane estimation using histograms of depths with fixed bandwidth. This helps in estimating the depth around corner features as well. (3) Then the algorithm estimates the plane that fits the feature using the triangulation method. However depth estimation for the features on the road includes another preemptive processing, which includes RANSAC for plane fitting. After performing the above steps, the next step includes frame to frame odometry estimation using perspective-n-point problem. Besides the procedures described above, there are certain steps

that needs to addressed including strategies to select the data in order to increase efficiency and robustness. It is quite important to select only few important landmark features, since there could be many in a dynamic environment that would increase computation complexity dramatically. For one of these reasons, the landmarks are classified into near, middle and far. Finally, it uses bundle adjustment on these detected features to estimate the ego motion (aka motion estimation of a camera system). This approach was entirely evaluated on KITTI dataset. The estimated trajectories are precise with low drift but it does not solve the loop closure problem. On the KITTI datasets, LIMO has the translation error of 0.93% and a rotation error of 0.00026 deg/meter, which has proved to be a significant contribution to the robotics community.

A different method proposed in [42], [45] deserves attention. Here, the authors put forward a method that utilizes depth data to estimate the camera motion. It also uses bundle adjustment to refine the motion estimation. At the time of release of this method, it was ranked as the first in the KITTI benchmark visual odometry methods. As a first step or block, visual features are detected and tracked. Depth images, which could be from RGB-D camera or from the point clouds, are registered in the depth registration block using the estimated motion. The final step is called the frame-to-frame motion estimation, which uses feature as input acquired using the sequence of images, and then these features are fed to the bundle adjustment procedure. The results were evaluated on the KITTI dataset where in the urban environment the relative mean position error was 1.05%, and in the highway it was 1.86%.

Another work by [35] offers a novel technique called Simultaneous Trajectory Estimation and Mapping (STEAM). This technique trains a Gaussian process model using the ground truth. The input to this system is a well detected features extracted from the point clouds, and the output of the system is the predicted poses that are computed using the estimator and the ground truth. In a more deeper level, this algorithm starts with Lidar point cloud down sampling, where the heavy data of point clouds is reduced to sparse points called key points using normalized intensity values. A point can be selected as a key point or not if it satisfies certain conditions based on the their proposed algorithm. Then these sparse point clouds are matched based on Euclidean distance. For estimating the trajectory, they implement the STEAM framework in which continuous time trajectory is estimated as Gaussian process regression. The authors have also mentioned a significant point that, for continuous-time trajectory estimation, the Gaussian regression problem is quite different from predicting odometry data. In order to reduce the errors in the odometry, the algorithm calculates the pose change from frame to frame and then compares it against the ground truth [5], [10], [19], [26], [36], [40]. In Gaussian process modelling, the model is learned from the noisy observations. So it becomes significant to select the features to detect in order to build a correct model. The results were evaluated on a KITTI dataset and the overall error from all the path segments was 1.16%.

A different approach to the SLAM problem is the Closet Probability and Feature Grid SLAM (CPFG-SLAM) [19], which has proposed a technique for localizing an unmanned vehicle in the off-road environment. In essence, it combines the features of the point cloud with probability and the occupancy probability of the grid map. Expected Maximization (EM) is further used to build the optimization function to match between point cloud and grid map. This technique comprises of 3 steps: data pre-processing, pose estimation, and updating feature grid map. Data pre-processing constitutes the filtering and classification of the point cloud. Pose estimation comprises of estimating the pose and the position by matching point cloud to the map. Finally, updating the point cloud features consists of extracting point cloud features and updating the probability of the grid. Later on the EM algorithm is performed using Levenberg-Marquadt algorithm. Despite high localization accuracy, this algorithm is not robust against dynamic environments, and also it does not solve the loop closure problem.

Another approach, by previously acclaimed authors [42], have proposed a real time monocular odometry, which is enhanced by depth data. This method is worth mentioning since it estimates depth from camera motion using sparse depth data too. It achieves this result by a method called triangulation that uses previously estimated motion and features from the image for which depth data is unavailable. Later on, it uses bundle adjustment to refine the estimated motion. Firstly, it tracks visual features from the images. Visual features are computed using Harris corner detection algorithm and is tracked using Kanade Lucas Tomasi (KLT) method [38]. Next, it uses the depth data (either from RGB-D camera or a Lidar) to register the point clouds with the depth using the estimated motion. The frame to frame motion estimation is done using bundle adjustment whose inputs are the features extracted from the sequence of images. One interesting thing to note in this algorithm is that it uses both known and unknown depths of features in order to estimate the odometry of the camera.

One of the novel techniques that demands attention is from [36], which uses a learning approach. This technique essentially trains a Gaussian process regression model using data with ground truth. All the high level features that are derived from the Lidar point clouds are used as input, and the predicted biases between poses from the estimator, and the ground truth is the output of the system. However, the whole process is divided into a number of steps. First, the point clouds is downsampled to represent only the keypoints or the well featured points in the point cloud. In this step it calculates the eigen values of the matrix, which represent the k-nearest neighbours of the point on the point clouds and sets a threshold through some functions to classify it a key feature point. It uses libpointmatcher library to do so [29]. Secondly, for two such downsampled point clouds, the point clouds are matched based on their Euclidean distance. It uses libnabo for matching [13]. Thirdly, it uses the STEAM framework, as discussed previously by [35], for trajectory estimation. However, only odometry section of the STEAM framework is implemented here. More information about how the error is predicted and corrected in the paper itself, in which most of it is derived from the STEAM framework. On the KITTI datasets, it performs relatively better, but not as good as IMLS and LOAM algorithms as discussed earlier. Based on the authors, since this is strictly odometry, it does not solve loop closure problem or reducing the drift in the odometry. Our understanding is that this algorithm has the potential to use deep learning framework for better odometry estimation

and correction rather than using a Gaussian prediction model.

Another novel approach for localization was done by [3] where a surfel based map is used. The changes in the robot pose can be estimated by the data association between the current scan of the Lidar and the model view from the surface map. This technique is also called as Surfel based Mapping (SuMa), which builds globally consistent maps. In addition to that, surfel allows us to represent large scale environment and also maintains detailed geometric information of the point clouds. Based on the current rapid development in the computation, rendering surfels is relatively fast. Odometry is computed using frame-to-model ICP with point-to-plane error metric. The error is minimized using Gauss-Newton minimization algorithm. This algorithm has been evaluated on KITTI dataset, which shows that the average rotational error of 0.00032 deg/m and a translational error of 1.4%.

B. Stereo Based Odometry

Perhaps the current state of the art algorithm for the stereo visual odometry is the SOFT-SLAM [11] that relies on feature tracking based algorithm. It builds a feature based pose graph and then optimizes it by running it in 2 separate threads. One is odometry thread, and the other is the mapping thread, which allows it to support large loop closing and global consistency. It achieves good localization with the use of featured visual odometry compared to the use of bundle adjustment, which is computationally very expensive. Unlike other algorithms like ORB-SLAM2, SOFT-SLAM algorithms are more deterministic (e.g., it results in the same output for the same dataset.)

Among the popular ones, we would like to mention the contribution of Large Scale Direct monocular SLAM (LSD-SLAM) [14]. LSD-SLAM has been one of the most popular SLAM technique. While most of the visual SLAM algorithms are based on features extracted from the images, LSD-SLAM algorithm is feature less algorithm, which allows us to build large-scale consistent maps of the environment in addition to tracking the motion of the camera. The reason behind this is that the features being used in most SLAM algorithms is completely dependent on the type of features being extracted, which in larger complex environment can be different. In this algorithm, the global map is represented as a pose graph, which consists of key frames as vertices, and the 3D similarity transforms as edges. The classic LSD-SLAM mainly has a 3-step process: tracking, depth map estimation, and map optimization. The tracking section tracks new image frames, which allows to estimate the rigid body frame with respect to current key frame. Depth estimation uses tracked frames to refine the current keyframe. The depth map generated after the depth map estimation block is fed into the global map using map optimization component.

The above work of LSD-SLAM was also extended to stereo camera [14]. It is based on almost the same technique, but the authors have exploited the use of stereo camera setup as well. In essence, the depth estimation is done concurrently in 2 setups. One is from the stereo camera setup with a fixed baseline, and the other is from the multi-view stereo established from the camera motion. The advantage of having a stereo with a fixed baseline setup is that it avoids the scale drift, which typically occurs in monocular LSD-SLAM. It also handles sudden illumination changes in the image frames using direct image alignment. This algorithm has been evaluated on KITTI dataset, and it is still one of the most popular odometry estimation algorithm with overall Root Mean Square Error (RMSE) of 1.21%.

Another stereo approach, given by [21], is based on Exactly Sparse Delayed State Filter (ESDSF) and EKF on Lie groups (LG-EKF) [22]. This algorithm preserves the state space geometry by representing it as a algebraic Lie group. Since the approach is based on ESDSF, which is derived from Extended Information Filter, its main advantage is that it uses sparse information matrix. One of the major features of this method is that it uses a novel ESDSF on a Lie groups, which not only presents all the advantages of classical ESDSF but also holds the state space geometry using Lie groups. This algorithm was evaluated on KITTI dataset and has been compared to various other SLAM algorithms like ORB-SLAM2, Stereo-Parallel Tracking and Mapping (S-PTAM), and Stereo LSD-SLAM (S-LSD SLAM). It has shown improvement in the odometry from most of the popular stereo based SLAM algorithms.

An approach by [6] presents an iterative 2-stage process for frame-to-frame feature based odometry estimation. This algorithm attempts to analyze the characteristics of optical flows and re-projection errors that are generated from the 6-DOF motion. They have justified the re-projection error that us generated from the optical flow algorithm, which is dependent on the coordinate of the features.

One of the algorithms that uses the direct visual odometry was proposed by [45]. This algorithm attempts to solve the problem of getting stuck at local optima at large displacements. It is done by dual Jacobian optimization in fusion with multi-scale pyramidal scheme. In addition to this, it introduces new features based on gradients, which is robust to illumination changes. Finally, a joint odometry is proposed to incorporate more information from last frame to previous key frames.

Stereo algorithms that tracks key points and selects effective frames is accomplished by another technique called Selective SLAM (SSLAM) [4]. The basic idea in this approach is that the error in localization or pose estimation is because of uncertainty of 3D points. This uncertainty is higher for distant points. One key feature is that it does not require any loop closure or bundle adjustment. It uses Harris corner detector to detect the features and Gradient Location and Orientation Histogram (sGLOH) descriptor to match them.

Given the complexity of the graph based SLAM, one approach, that attempts to simplify the implementation of graph based SLAM and also challenges other state of the art SLAM algorithm, is the ProSLAM [30]. It's main goal is to use the stereo images to generate a 3D map. As it is quite evident that landmarks are essential since it has gotten descriptors and also allows to detect loop closures. In this approach, the node of the graph represents the pose (rotation and translation component) and the edge represents the spatial constraints. It is generated either by tracking the camera motion or by aligning local maps acquired at distant times, which also leads to loop closure that is again helpful for re-localization. The whole approach is taken into 4 steps: (1) Frame point generation, which takes the pair of stereo images and generates 3D points; (2) Position tracking, which estimates the relative motion of the

camera from two subsequent image pairs acquired at different times as the camera moves; (3) Map management, which collects all the acquired map (3D point clouds) obtained from the trajectory and represents it in a compact form; (4) Re-localization, which compares each acquired point cloud from the previously acquired map and corrects the pose as well as the global map of the environment. This algorithm was evaluated on the KITTI dataset and has showed less than around 1% of translational error for most of the sequences. It has outperformed popular SLAM techniques like S-PTAM [27], [28] and LSD-SLAM but is on a competitive level on the ORB-SLAM2 [25].

Most of the SLAM techniques face the challenge of correcting the scale drift from the acquired images. Research by [33], has attempted to resolve this issue. This work uses monocular camera, and it estimates the ground plane using a novel cue combination framework. It achieves results comparable to stereo algorithms and could be carried out over long data sequences. From every monocular images, it uses sparse features, that are acquired from the stereo images. At the same time it performs object detection as well. It provides a model-learning approach from the training data, which is related to covariances of the observation cues. Based on the KITTI dataset evaluation, this technique has outperformed VISO2 [16], [20] for both monocular and stereo camera. The above method has attempted to build a bridge between monocular and stereo structure from motion in addition to correct the scale drift. From the same researchers, another work that deserves attention was parallel visual odometry estimation [34]. The approach is to use multi-threading for scenes with large motions and rapidly changing images. It uses three or more CPU parallel threads, and across all the threads the system estimates the pose using 3D-2D correspondences, which is again followed by bundle adjustment.

Using features and descriptors has been one of the most popular approaches to perform localization. [23] uses feature detection algorithms like Oriented fast and Rotated Briefs (ORB) to detect the features from the image sequences and computes feature descriptors using the Fast Retina Key point (FREAK) algorithm. This approach has been popularly known as the Circular Freak ORB (CFORB) algorithm. One of the key advantages it presents is that since it uses ORB features, it is invariant to both rotation and scale changes. It has also highlighted that it is invariant to environment to uneven terrain changes. Another thing to highlight is that it uses two geometrical constraints in order to remove invalid geometrical feature matches, which was not implemented in common visual odometry algorithm. On KITTI benchmark dataset, this has shown an average translational error of 3.73% and a rotation error of 0.0107deg/rad. This approach has been tested on indoor environments as well, which performs slightly better than the outdoor environment and in addition, it also performs well in heavy textured environment.

III. Security in Autonomous Driving

Although autonomous vehicles as projected, may lead to safer roads, reduced congestion and solve the parking problem, it still faces the challenge of security over the network. Since autonomous cars heavily rely on digital systems or computers, it is very likely to have communication protocols and frameworks employed in the system. At this point there are two levels of networking among autonomous vehicles: Vehicle to Vehicle - inter vehicle networking around the vicinity of the vehicle in a local area; and Vehicle to Infrastructure - networking between vehicle and infrastructure system. Sharing vital information among vehicles like the speed, location and activity could assist in efficient navigation on the roads. However this system of complex architecture across wide networks can be attacked by various techniques, broadly classified into 2 categories: Active and Passive attacks [41].

Passive attacks are not intended to change the functionality of the system, rather it is done by a potential attacker to acquire confidential information about the system. A simple scenario could be eavesdropping where an attacker could obtain information by intercepting the data traffic. Even if encryption is used, a successful decryption can be counted as a passive attack. Another scenario could be to analyze the traffic signal, which allows the attacker to understand certain properties like information transaction based on duration, timing, bandwidth and number of participants in the traffic [41].

Active attacks are way more inclusive in terms of functionality and changing the system as well. There are various ways where a system could be hacked like man-in-the-middle attack, Denial of Service (DoS) attack and Replay attack. In man-in-the middle attack [31], [32], the attacker can fraudulently get access to the system. In addition to this, a man-in-the-middle attack send an acknowledgement to the sender that it is a authorized user thereby deceiving the whole system. In a replay attack, the attacker can observe the data traffic and replay a previous message and could force the system into an unstable state or could request for further data for attacking at different levels of security. DoS simply means that the system has been compromised, which could be done by disabling the communication services or jamming the communication channels.

With respect to autonomous cars, there has been demonstrations by researches who have successfully attempted to gain a complete control of the autonomous system including disabling the brakes, stopping the engine, locking the doors etc., and most importantly completely ignoring the driver input signals. Most of the autonomous cars follow the Controller Area Network (CAN)-protocol. However this has presented various vulnerabilities in the communication in the automotive industry. Primarily being, CAN has broadcast nature, and it sends packets to all the nodes in the network, which might allow the attacker to insert malicious component easily. CAN is also vulnerable to DoS attacks. In addition, CAN has no proper authentication mechanism. Anyone can send packet to any node quite easily. It is quite important to mention that any attacker can attack an autonomous system easily, which makes it very essential for further research improvements in the autonomous driving industry [2], [39], [41].

IV. Conclusions

This paper has briefly described various SLAM techniques in relation to autonomous driving evaluated especially in KITTI dataset. We have attempted to classify the SLAM techniques in both Lidar-based odometry and Stereo-based odometry. In the end, we have also attempted to describe the security vulnerabilities in the autonomous driving systems. We have described various types of attacks that have been

introduced and demonstrated by various researchers, which makes it a popular topic for future research. We intend to focus our future research on improving the loop closure detection robustly using graph based SLAM techniques with a primary target on evaluating our approach on KITTI dataset. In addition to it, we also intend to focus on including and validating deep learning approach to SLAM, since it probably provides promising results for data analysis.

REFERENCES

[1] KITTI Dataset, available at. http://www.cvlibs.net/datasets/kitti/. Accessed: 2019-01-10.

[2] Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 77:167 – 181, 2015.

[3] J. Behley and C. Stachniss. Efficient surfel-based slam using 3d laser range data in urban environments. In *Robotics: Science and Sys.*, 2018.

[4] F. Bellavia, M. Fanfani, F. Pazzaglia, and C. Colombo. *Robust Selective Stereo SLAM without Loop Closure and Bundle Adjustment*. Springer, 2013.

[5] M. Buczko and V. Willert. How to distinguish inliers from outliers in visual odometry for high-speed automotive applications. In *IEEE Intel. Vehicles Symposium*, 2016.

[6] M. Buczko, V. Willert, J. Schwehr, and J. Adamy. Self-validation for automotive visual odometry. In *IEEE Intel. Vehicles Symp.*, 2018.

[7] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. on Robotics*, 32(6):1309–1332, Dec 2016.

[8] J. Civera, D. Glvez-Lpez, L. Riazuelo, J. D. Tards, and J. M. M. Montiel. Towards semantic slam using a monocular camera. In *IEEE/RSJ Intern. Conf. on Intel. Robots and Systems*, pages 1277–1284, Sep. 2011.

[9] D. Connell and H. M. La. Extended rapidly exploring random treebased dynamic path planning and replanning for mobile robots. *Intern. J. of Advanced Robotic Systems*, 15(3):1729881418773874, 2018.

[10] I. Cvii and I. Petrovi. Stereo odometry based on careful feature selection and tracking. In *European Conf. on Mobile Robots (ECMR)*, 2015.

[11] I. Cvii, J. esi, I. Markovi, and I. Petrovi. Soft-slam: Computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles. *Journal of Field Robotics*, 35(4):578–595, 2018.

[12] J. Deschaud. Imls-slam: Scan-to-model matching based on 3d data. In *2018 IEEE Intern. Conf. on Robotics and Automation (ICRA)*, pages 2480–2485, May 2018.

[13] J. Elseberg, S. Magnenat, R. Siegwart, and A. Nüchter. Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration. *J. of Software Engineering for Robotics (JOSER)*, 3(1):2–12, 2012.

[14] J. Engel and J. St˙ Large-scale direct SLAM with stereo cameras.

[15] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[16] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intel. Vehicles Symposium (IV)*, 2011.

[17] J. Graeter, A. Wilczynski, and M. Lauer. Limo: Lidar-monocular visual odometry. *arXiv preprint arXiv:1807.07524*, 2018.

[18] G. Grisetti, R. Kmmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based slam. *IEEE Trans. on Intel. Transportation Systems Magazine*, 2:31–43, 12 2010.

[19] K. Ji and J. G. G. X. J. Q. T. Y. Huiyan Chen, Huijun Di. Cpfg-slam:a robust simultaneous localization and mapping based on lidar in off-road environment. In *IEEE Intel. Vehicles Symposium (IV)*, 2018.

[20] B. Kitt, A. Geiger, and H. Lategahn. Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *Intel. Vehicles Symposium (IV)*, 2010.

[21] K. Lenac, J. esi, I. Markovi, and I. Petrovi. Exactly sparse delayed state filter on lie groups for long-term pose graph slam. *The Intern. J. of Robotics Research*, 37(6):585–610, 2018.

[22] K. Lenac, J. esi, I. Markovi, and I. Petrovi. Exactly sparse delayed state filter on lie groups for long-term pose graph slam. *The Intern. J. of Robotics Research*, 37(6):585–610, 2018.

[23] D. J. Mankowitz and E. Rivlin. CFORB: Circular FREAK-ORB Visual Odometry. *arXiv preprint arXiv:1506.05257*, 2015.

[24] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans. on Robotics*, 31(5):1147–1163, 2015.

[25] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Trans. on Robotics*, 33(5):1255–1262, 2017.

[26] F. Neuhaus, T. Koss, R. Kohnen, and D. Paulus. Mc2slam: Real-time inertial lidar odometry using two-scan motion compensation. In *German Conf. on Pattern Recognition*. Springer, 2018. To appear.

[27] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, and J. Jacobo Berlles. S-PTAM: Stereo Parallel Tracking and Mapping. *Robotics and Autonomous Systems (RAS)*, 93:27 – 42, 2017.

[28] T. Pire, T. Fischer, J. Civera, P. De Cristóforis, and J. Jacobo berlles. Stereo Parallel Tracking and Mapping for robot localization. In *Proc. of the Intern. Conf. on Intel. Robots and Systems (IROS)*, pages 1373–1378, September 2015.

[29] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat. Comparing ICP Variants on Real-World Data Sets. *Autonomous Robots*, 34(3):133–148, Feb. 2013.

[30] D. Schlegel, M. Colosi, and G. Grisetti. ProSLAM: Graph SLAM from a Programmer's Perspective. In *2018 IEEE Intern. Conf. on Robotics and Automation (ICRA)*, pages 1–9, 2018.

[31] A. Singandhupe, H. M. La, and D. Feil-Seifer. Reliable security algorithm for drones using individual characteristics from an eeg signal. *IEEE Access*, 6(1):2–12, 2018.

[32] A. Singandhupe, H. M. La, D. Feil-Seifer, P. Huang, L. Guo, and M. Li. Securing a uav using individual characteristics from an eeg signal. In *IEEE Intern. Conf. on Systems, Man, and Cybernetics (SMC)*, 2018.

[33] S. Song and M. Chandraker. Robust scale estimation in real-time monocular sfm for autonomous driving. In *CVPR*, Columbus, Ohio, USA, 24-27, 2014.

[34] S. Song, M. Chandraker, and C. C. Guest. Parallel, real-time monocular visual odometry. In *ICRA*, Karlsruhe, Germany, May 6-10, 2013.

[35] T. Y. Tang, D. J. Yoon, and T. D. Barfoot. A white-noise-on-jerk motion prior for continuous-time trajectory estimation on se (3). *arXiv preprint arXiv:1809.06518*, 2018.

[36] T. Y. Tang, D. J. Yoon, F. Pomerleau, and T. D. Barfoot. Learning a Bias Correction for Lidar- only Motion Estimation. *15th Conf. on Computer and Robot Vision (CRV)*, 2018.

[37] K. Tateno, F. Tombari, I. Laina, and N. Navab. CNN-SLAM: real-time dense monocular SLAM with learned depth prediction. *CoRR*, abs/1704.03489, 2017.

[38] C. Tomasi and T. Kanade. Detection and tracking of point features. *Inter. J. of Computer Vision*, 9(3):137–154, 1991.

[39] A. M. Wyglinski, X. Huang, T. Padir, L. Lai, T. R. Eisenbarth, and K. Venkatasubramanian. Security of autonomous systems employing embedded computing and sensors. *IEEE Micro*, 33(1):80–86, Jan 2013.

[40] N. Yang, R. Wang, J. Stueckler, and D. Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *European Conf. on Computer Vision*, September 2018.

[41] E. Yadereli, C. Gemci, and A. Z. Akta. A study on cyber-security of autonomous and unmanned vehicles. *The J. of Defense Modeling and Simulation*, 12(4):369–381, 2015.

[42] J. Zhang, M. Kaess, and S. Singh. Real-time depth enhanced monocular odometry. In *IEEE/RSJ Intern. Conf. on Intel. Robots and Systems (IROS)*, Chicago, IL, Sept. 2014.

[43] J. Zhang and S. Singh. LOAM: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems Conf.*, Berkeley, CA, July 2014.

[44] J. Zhang and S. Singh. Visual-lidar odometry and mapping: Low drift, robust, and fast. In *IEEE Intern. Conf. on Robotics and Automation (ICRA)*, Seattle, WA, May 2015.

[45] J. Zhu. Image gradient-based joint direct visual odometry for stereo camera. In *Intern. Joint Conf. on Artificial Intelligence, IJCAI*, 2017.