MCC-EKF for Autonomous Car Security

Ashutosh Singandhupe, Hung Manh La

***Abstract*—This work attempts to answer two problems. (1) Can we use the odometry information from two different Simultaneous Localization And Mapping (SLAM) algorithms to get a better estimate of the odometry? and (2) What if one of the SLAM algorithms gets affected by shot noise or by attack vectors, and can we resolve this situation? To answer the first question we focus on fusing odometries from Lidar-based SLAM and Visual-based SLAM using the Extended Kalman Filter (EKF) algorithm. The second question is answered by introducing the Maximum Correntropy Criterion - Extended Kalman Filter (MCC-EKF), which assists in removing/minimizing shot noise or attack vectors injected into the system. We manually simulate the shot noise and see how our system responds to the noise vectors. We also evaluate our approach on KITTI dataset for self-driving cars.**

***Keywords*—extended kalman filter; maximum correntropy criterion; SLAM; autonomous car security**

I. Introduction

SLAM researchers have developed numerous algorithms that are primarily based on visual/geometry cues of the surrounding environment. Given SLAM's widespread applications in various autonomous systems, it has gained significant attention in the research community [3], [35]. More importantly, SLAM is an essential component, which enables the self-driving capability in modern car systems [26]. Navigating an autonomous system in Global Positioning Systems (GPS) denied environments is the primary drive for SLAM research. Even though GPS improves localization, numerous SLAM techniques are focused on improving localization without using GPS.

Probabilistic estimation techniques like Kalman Filters (KF) [13] are the starting point of understanding and implementing modern SLAM systems. It's variants like Extended Kalman Filters (EKF) [9], [14], [16], [17], and Unscented Kalman Filters (UKF) were later on proposed for non-linear SLAM systems. Other approaches like Particle filters have also shown significant improvement in SLAM research [3]. Another unique approach that addresses the problem of filtering based approaches is the graph-based SLAM [35]. Here the robot pose is modeled as a node/vertex in a graph representation, and the edges represent the errors in measurements from various sensors. Eventually, the process results in generating a pose graph structure and the resulting error can be minimized by using mathematical optimization techniques like Gauss-Newton/Levenberg–Marquardt. Popular SLAM techniques like Oriented fast and Rotated Briefs-SLAM (ORB2-SLAM) [23], [24] uses the graph-based approach for localization. Besides the common convention, the advent and the high success rate of deep learning have given birth to Convolutional Neural Networks (CNN), which is a unique direction in deep learning-based SLAM research. Subsequently, quite interesting results were observed especially with the work on CNN-SLAM [39]. The experiments have shown that localization can be achieved from a pair of images acquired by a moving robot through CNN-based deep learning framework. Despite promising results from CNN-SLAM, this approach invites a few challenges that need to be addressed. Deploying high-end GPU systems on robotic embedded systems capable of solving complex deep learning problems is still a challenge. Moreover learning complex dynamic environments is still a challenge in modern SLAM systems. This requires heavy computation if it uses a deep learning framework. From the various techniques introduced in SLAM, one can observe that SLAM inclines to combine various fields like signal processing, deep learning (CNN-SLAM) and more significantly computer vision [34], [38], [39], [42].

Despite the flood of numerous SLAM algorithms that have been proposed so far, very few of them address the problem of securing the autonomous system in case it gets attacked. Numerous incidents have been reported, where researchers have attacked the autonomous car systems (for example Tesla) that made the car change it's naturally estimated trajectory [2]. This has raised a serious concern in modern autonomous systems and needs to be addressed [36], [37], [40]. Although cyber-security experts have proposed various solutions in solving those issues, these security/hacking problems are still remaining open and challenge. It is important to note that any system can potentially be attacked. Now, if a SLAM system is vulnerable, it becomes a challenge to deploy it in real-time scenarios. So, can a SLAM system be secured by itself? which means can a SLAM system be designed in a way where it can detect an attack/outliers by itself to avoid the change in it's natural estimated trajectory?

Our work is focused on solving this SLAM security problem by building a self-secure SLAM framework that can detect potential outliers/attacks. We propose to use the Maximum Correntropy Criterion - Extended Kalman Filter (MCC-EKF) approach in our framework and show through the results of how our approach avoids attacks. The framework that we build is based on using the odometry from two SLAM methods and fusing them using MCC-EKF, which results in the overall estimation of the autonomous system. This also answers the question - Can we use two odometry from different SLAM methods and get a better estimate of the trajectory?. The results show that we can improve the overall trajectory. But initially,

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. This work is also partially supported by the U.S. National Science Foundation (NSF) under grants NSF-CAREER: 1846513 and NSF-PFI-TT: 1919127. The views, opinions, findings and conclusions reflected in this publication are solely those of the authors and do not represent the official policy or position of the NASA and NSF.

Video of the MCC-EKF implementation on KIITI dataset is available at: https://youtu.be/bKzN8NdQnaE

The source code of MCC-EKF in ROS is available at the ARA lab's github: https://github.com/aralab-unr/MCC-EKF-SLAM

978-1-7281-5237-0/20/$31.00 ©2020 IEEE
DOI 10.1109/IRC.2020.00056

we introduce the idea of MCC-EKF using the Gaussian Kernel correntropy function and discuss the mathematical interpretation of it [11]. The prime motivation for our work is to resolve a situation when the system gets attacked, that forces the autonomous system to change its natural estimated trajectory [41] [8]. In several examples, we have seen how an attacker can attack an autonomous system and change its trajectory to the attacker's own desired location [2]. We propose to use the MCC-EKF SLAM algorithm in our system, which has the potential to give a solution to this problem. We use this approach in our system as well as evaluate our approach in the popular KITTI dataset [1] and discuss the results.

The remaining paper is organised as follows: Section II introduces the Correntropy Kalman Filter and its underlying concepts. Section III describes our implementation of MCC-EKF algorithm. The results and evaluation of our proposed methodology is discussed in IV.

II. Kalman Filter with Correntropy

1) Correntropy: We will first discuss the correntropy criterion since it has gained popularity in various fields such as pattern recognition, machine learning and designing the filter in the presence of non-Gaussian noise. It has proved beneficial to remove large outliers. Correntropy is essentially a similarity measure of two scalar random variables. It can be mathematically represented as :

$$V_\sigma(X,Y) = E[\kappa_\sigma(X-Y)]. \tag{1}$$

This also refers to in common literature as a cross-correntropy of two scalar random variables X and Y [11] [4] [8]. In Equ. (1), $E[.]$ refers to the expectation of the variable, and κ_σ denotes the kernel function. In our approach, we use the Gaussian Kernel function where the correntropy can be rewritten as:

$$V_\sigma(X,Y) = \frac{1}{N}\sum_{i=1}^{N} G_\sigma(x_i - y_i) \tag{2}$$

where x_i and y_i are N random samples drawn from X and Y. Equ. (2) plays a key role in our work where,

$$G_\sigma(x_i - y_i) = exp(-\frac{\|x_i - y_i\|^2}{2*\sigma^2}) \tag{3}$$

and σ is the bandwidth or the kernel size of the Gaussian kernel. From Equ. (3) it becomes evident that if $X = Y$ Gaussian Correntropy is maximum, and the Gaussian correntropy function is positive and bounded [11] [4] [8]. The next subsection describes how this can be used with Kalman Filtering.

2) Correntropy with Kalman Filtering: Kalman filter for state estimation is given as follows:

$$x_k = F * x_{k-1} + w_k \tag{4}$$

$$y_k = H * x_k + v_k \tag{5}$$

where $x_k \varepsilon R^n$ is the state vector, and $y_k \varepsilon$ is the measurement vector. w_k is the system process noise, and v_k is the measurement noise, and both are assumed to be zero mean. F and H are the system matrix and the observation matrix, respectively. We represent the associated covariance matrix to be Q_k and R_k for the system process noise and the measurement noise, respectively. Kalman filter operates as a weighted least squares approach where and objective function, J, is defined as:

$$\begin{aligned} J = \tfrac{1}{2} * (y_k - H * \hat{x}_k)^T * R_k^{-1} * (y_k - H * \hat{x}_k) + \\ \tfrac{1}{2} * (\hat{x_k} - F * \hat{x}_{k-1})^T * P_{k|k-1}^{-1} * (\hat{x}_k - F * \hat{x}_{k-1}) \end{aligned} \tag{6}$$

The KF can be derived by solving

$$\frac{\partial J}{\partial \hat{x}_k} = 0. \tag{7}$$

Then the KF after initialization of the state variables is given as:

$$P_{k|k-1} = F P_{k-1|k-1} F^T + Q_k \tag{8}$$

$$K_k = P_{k|k-1} H^T (H P_{k|k-1} H^T + R_k)^{-1} \tag{9}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - H\hat{x}_k^-) \tag{10}$$

$$P_{k|k} = (I - K_k H) P_{k|k-1} (I - K_k H)^T + K_k R_k K_k^T \tag{11}$$

where $K_k \in \mathbb{R}^{n \times m}$ is the Kalman gain. $\hat{x}_k^-$ is the priori estimate of the state x_k. It is based on measurements up to and including time $k-1$, and has covariance $P_{k|k-1}$. $\hat{x}_k$ is the posteriori estimate of the state x_k, and it is based on measurements up to and including time k and has covariance $P_{k|k}$.

Now, with respect to the Correntropy Kalman Filter the objective function in Equ. (6) becomes,

$$\begin{aligned} J = G_\sigma(||y_k - H * \hat{x}_k||_{R_k^{-1}}) + \\ G_\sigma(||\hat{x}_k - F * \hat{x}_{k-1}||_{P_{k|k-1}^{-1}}). \end{aligned} \tag{12}$$

The derivation for this can be seen in [11] [4] [8], and the overall equations for the MCC-EKF can be written as follows:

For initialization:

$$\hat{x_0} = E[x_0] \tag{13}$$

$$P_0 = E[e_0 e_0^T] \tag{14}$$

where, $e_0 = x_0 - \hat{x}_0$ is the error term, and $E(.)$ is the expected value operation. Prior estimation can be calculated as

$$\hat{x}_k^- = F\hat{x}_{k-1} \tag{15}$$

$$P_{k|k-1} = F P_{k-1|k-1} F^T + Q_k. \tag{16}$$

The remaining state estimation can be calculated as:

$$L_k = \frac{G_\sigma(||y_k - H * \hat{x}_k||_{R_k^{-1}})}{G_\sigma(||\hat{x}_k - F * \hat{x}_{k-1}||_{P_{k|k-1}^{-1}})} \tag{17}$$

$$K_k = (P_{k|k-1}^{-1} + L_k * H^T * R_k^{-1} * H)^{-1} * L_K * H^T * R_K^{-1} \tag{18}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - H * \hat{x}_k^-) \tag{19}$$

$$P_{k|k} = (I - K_k * H) * P_{k|k-1} * (I - K_k * H)^T + K_k * R_k * K_k^T. \tag{20}$$

From the derivation of L_k, it can be seen that, if y_k is large (outlier measurement), G_σ approaches zero and so is L_k. So if L_k is zero then the Kalman Gain K_k is zero, which means

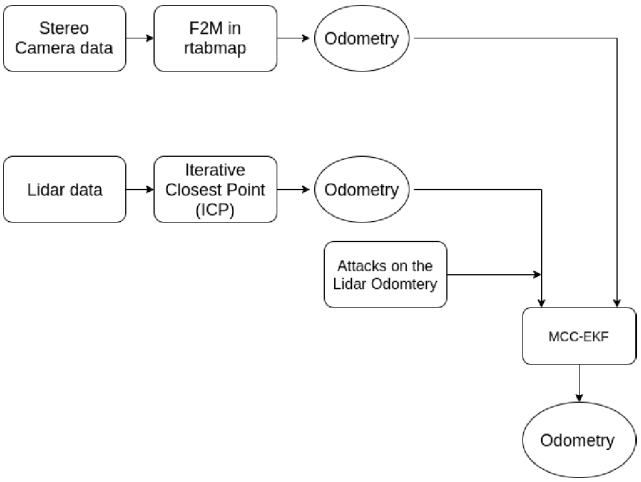


Fig. 1: System Architecture.

that the state update in Equ. (19) is updated by the state of the system as given by Equ. (15). The mathematical representation of the above equations can be written as:

$$\lim_{y_k \to \infty} L_k = 0 \tag{21}$$

$$\lim_{L_k \to 0} K_k = 0 \tag{22}$$

$$\lim_{K_k \to 0} \hat{x}_k^- = \hat{x}_k^- . \tag{23}$$

III. Proposed Methodology

Figure 1 gives the basic system architecture of our proposed approach. As mentioned in Section I, we use two different SLAM algorithms to get the odometry. We also use two different sensors to accomplish our goal. We use simulated Velodyne Lidar (VLP-32) and a simulated stereo camera in our setup. We have evaluated our system in a gazebo simulated environment from an open-source repository [31]. We attempt to evaluate how using the odometry information from the two SLAM methods [22] [10] to enhance the overall odometry of the autonomous system (e.g., car).

For calculating the Lidar odometry we use the Iterative closest Point (ICP) SLAM, which uses 3D lidar data for estimating the 6 DOF position of the Lidar sensor. For our purpose, we are using only the raw Lidar measurements. In this approach, at first the Lidar data is downsampled for reducing computational complexity. We use a voxel grid filter of size 0.2 to downsample the point clouds. Later on, it estimates the sensor pose using the ICP, which is applied iteratively in the consecutive frames. We use point-to-plane error metric for faster convergence. Every estimated pose is then fed to a pose graph approach to optimize the relative pose as well as to detect the loop closure. The results of this approach are shown in the result section. For the ICP parameters, we have set the maximum iterations to be 100 and the maximum corresponding ratio is set to be 0.01.

Using the stereo camera data we calculate the odometery using rtabmap's Frame-To-Map (F2M) strategy [22].

Our approach takes advantage of these two SLAM methods, and we feed their respective calculated odometry in our MCC-EKF framework. Here we also propose to introduce shot noises or attack vectors into the estimated Lidar odometry and see how our framework responds to the attacks. Let the attack vector at time k be a_k. So the measurement equation is updated as $y'_k = y_k + a_k$. Then Equ. (24) gets updated as:

$$L_k = \frac{G_\sigma(||y'_k - H * \hat{x}_k||_{R_k^{-1}})}{G_\sigma(||\hat{x}_k - F * \hat{x}_{k-1}||_{P_{k|k-1}^{-1}})}. \tag{24}$$

So when y'_k is large as defined by the kernel bandwidth, L_k is 0, which forces the Kalman Gain K_k to be 0. So the next updated state is upated by the system state as given in Equ. (15). This process, as one can see, can reject the attacks/outliers thus securing the system.

The MCC-EKF algorithm for our approach is given below.

Algorithm 1: MCC-EKF algorithm for autonomous system security

Result: Computed odometry from MCC-EKF SLAM
Lidar Odom (ICP) $\rightarrow$L$_o(x, y, z, r, p, y)$
Stereo Odom (F2M) $\rightarrow$S$_o(x, y, z, r, p, y)$
Initialization;
Lidar odom $\rightarrow$ (L$_o$)
Stereo odom $\rightarrow$ (S$_o$)
Compute x_0 from Equ. (13)
Compute P_0 from Equ. (14)
Prior Estimation;
Compute $\hat{x}_k^-$ from Equ. (15)
Compute $P_{k|k-1}$ from Equ. (16)
while *get L_o and S_o;*
do
  Compute L_k from Equ. (24)
  Compute Gain K_k from Equ. (18)
  Update state x_k from Equ. (19)
  Update $P_{k|k}$ from Equ. (20)
end

IV. Results

We evaluate our approach on two systems - the simulated gazebo system and the KITTI dataset. The Gazebo simulated system uses a Prius model with inbuilt Lidar and a camera system. The source code and the model description is available as open source [31]. Figure 2 shows the simulated gazebo environment for our work, and Figure 3 shows a zoom-in location with the prius car model that we use. We have edited the model in order to include a stereo system so that we can get the odometry from the stereo camera (using rtabmap's FrameToMap (F2M)).

Figure 4 shows one sequence of the data from KITTI dataset. Again, it is mentioned earlier that we are using two different SLAM algorithms - ICP for Lidar odometry calculation and FrameToMap Visual odometry calculation (from rtabmap) for stereo cameras, and we are injecting attacks on the Lidar odomtery and evaluate how the MCC-EKF responds. Generally, for our framework, we can use any two SLAM algorithms to output odometry.

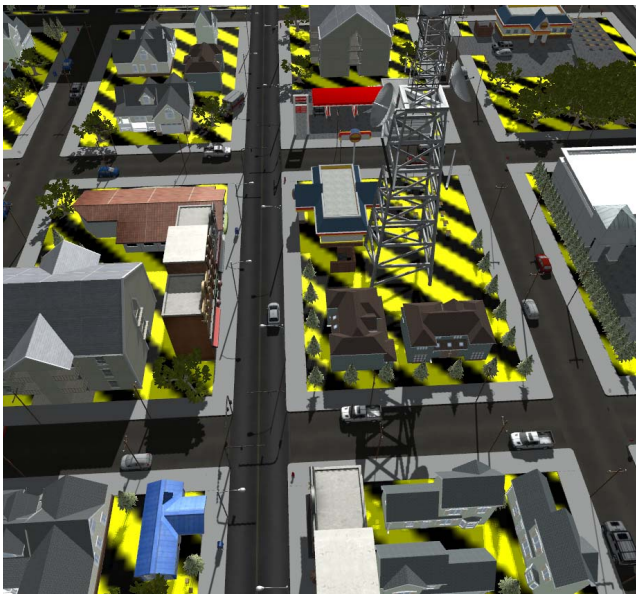


Fig. 2: Gazebo Simulation Environment.

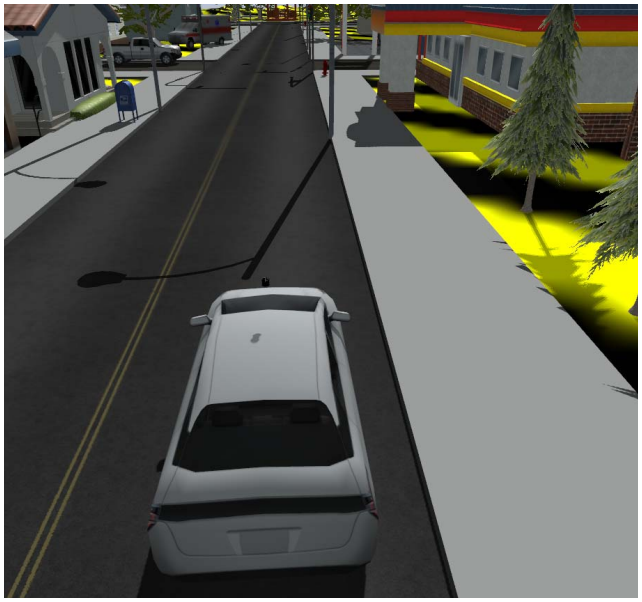


Fig. 3: Zoom-in at one location in Gazebo Simulation Environment.

For the Lidar odometry, we use ICP based approach to retrieve the odometry from Lidar. Initially, we downsample the point cloud using voxel grid filtering, and we use the Point-to-Plane error metric for faster convergence of the ICP algorithm. The sample result from our gazebo simulated model is shown in Figure 5. Figure 5(a) shows the Lidar odometry and the map. Figure 5(c) shows the Lidar trajectory with respect to the ground truth. The dotted line is the ground truth.

Using the stereo camera we calculate another set of odometry using Frame2Map in rtabmap. The sample result of the

(a)

(b)

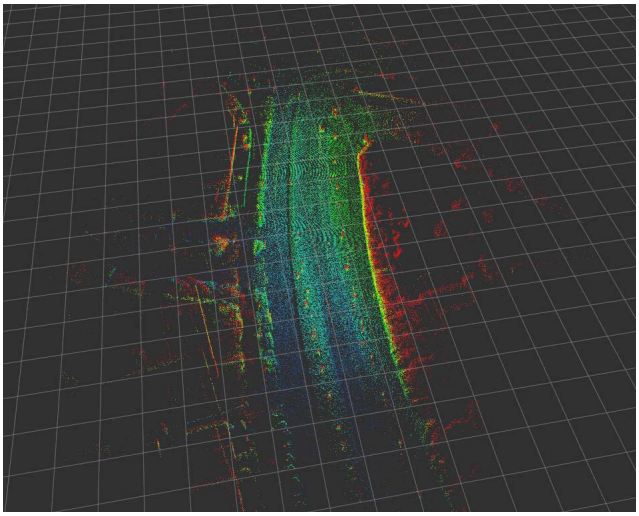


Fig. 4: KITTI dataset data (SEQ 11): (a)- Environment, and (b) its 3D Lidar map.

odometry as well as the mapping is shown in Figure 5(b) and Figure 5(d)

The next step involves using the odometries obtained from the above mentioned methods into our MCC-EKF framework. The combined odometries (Lidar odometry, Stereo odometry and the MCC-EKF odometry) for the example shown in Figure 5 is shown in Figure 6. In Table 1, this trajectory is referred as Trajectory 1.

Our initial query as mentioned earlier was, can we improve the odometry by using these two odometries obtained from different SLAM algorithms? In our experiment (Gazebo simulation) we compare the Root Mean Square Error (RMSE) values of individual trajectories with respect to the ground truth. Table I compares the RMSE values, which clearly shows that MCC-EKF performs better than the individual SLAM algorithms.

Another problem that we intended to solve was to avoid attacks (false injection of odometry values). This was the primary reason for our work presented here. We inject false values in the lidar odometry and test the response of our MCC-EKF approach. Table II indicates the RMSE values of the algorithms when false odometry is injected. We inject constant false values at random location in the lidar odometry, and we see that MCC-EKF can handle those attacks as compared to the use of the traditional EKF algorithm. The sigma parameter

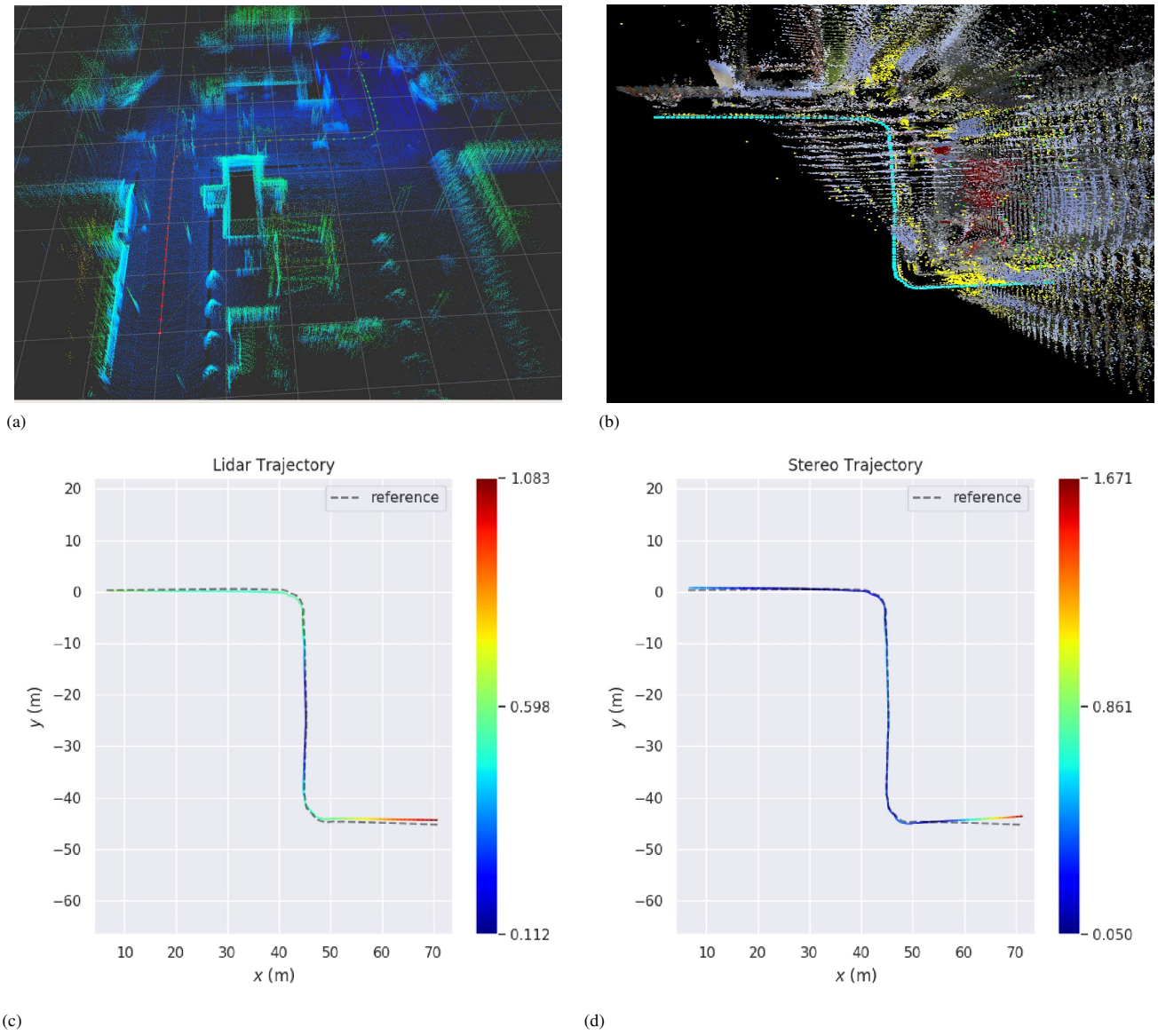


Fig. 5: (a, c) Lidar odometry and (b, d) Stereo odometry.

of the Gaussian kernel function G_σ plays an important role in MCC-EKF implementation. For our experiment, it is set to 10. Figure 7 shows the response of the MCC-EKF to the attacks on the Lidar data. In Figure 7 it is important to note that the trajectories are translated so that all the paths are displayed clearly. One can see that MCC-EKF does not get affected at places where the attacks were introduced.

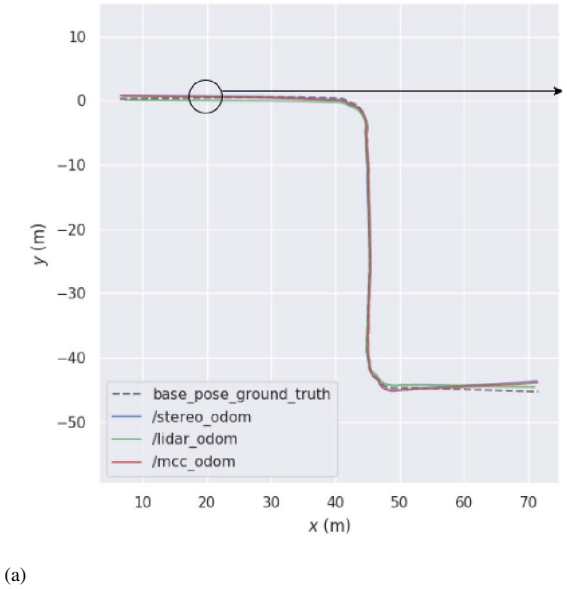
TABLE I: RMSE comparison

	Traj 1	Traj 2
Stereo (RMSE)1	0.478404	1.840472
Lidar (RMSE)	0.557076	0.505731
MCC-EKF (RMSE)	0.419823	0.49761

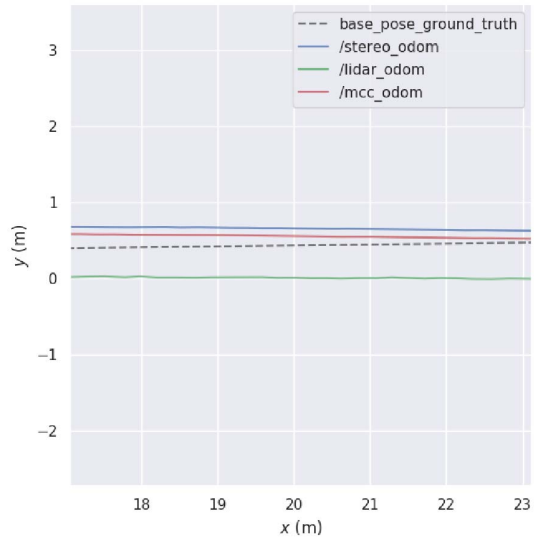
TABLE II: RMSE comparison after attack vectors

	Traj 1	Traj 2
Random attacks 1		
Normal EKF (RMSE)	3.624596	5.01472
MCC-EKF (RMSE)	0.420437	0.81037
Random attacks 2		
Normal EKF (RMSE)	4.624596	7.01472
MCC-EKF (RMSE)	0.43317	0.85132

As it is clear, MCC-EKF is a variant of the traditional EKF, and the MCC-EKF is robust to attacks or sudden outliers. So if we introduce attacks in the system (attacking Lidar odometry), the trajectory changes drastically. However using the MCC-EKF



(a)



(b)

Fig. 6: (a) Odometry trajectory of each method. (b) Zoom-in at one location.

approach, the attacks are rejected.

We have evaluated approach on the KITTI dataset. For the KITTI dataset, we have shown our results in two sequences. We have compared the results of MCC-EKF with normal EKF with and without attacks. The comparison of both the systems is shown in Figure 9 and 10. Figure 9 shows the normal EKF response with attacks while Figure 10 shows the MCC-EKF response with attacks. We can clearly see that the MCC-EKF can successfully eliminate the attacks, but the EKF can not. The source code is available on our ARA lab github: https://github.com/aralab-unr/MCC-EKF-SLAM

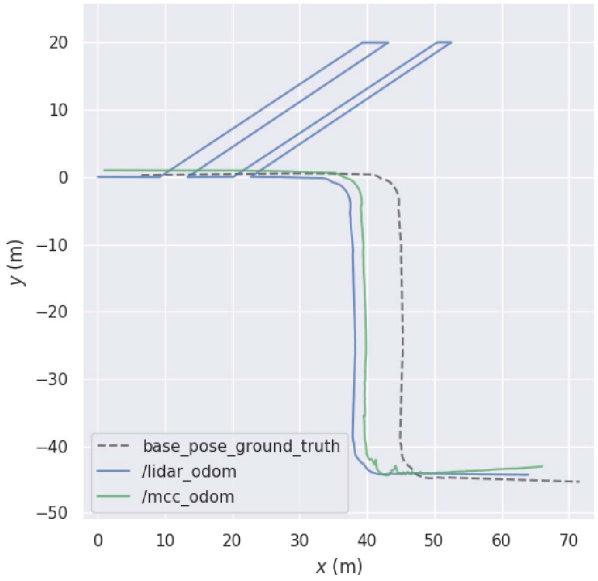


Fig. 7: MCC-EKF response to attacks on Lidar data.

TABLE III: RMSE comparison on KITTI dataset.

	SEQ 01	SEQ 05	SEQ 11	SEQ 27
Random attacks 1 EKF	2.04	2.42	1.475	3.193625
MCC-EKF	0.0198	0.0073	0.110	0.142083
Random attacks 2 EKF	3.79	7.0685	2.1875	4.1989
MCC-EKF	0.0198	0.092263	0.1842	0.1693

V. Conclusions

In this work, we have attempted to provide a self-secure solution to an autonomous system using the MCC-EKF approach. From the results we have shown how an autonomous system can be attacked by an attacker/hacker and change the system's naturally estimated trajectory and how even a simple injection of false positions can affect the overall trajectory of the autonomous system. We have also shown how the MCC-EKF approach can resolve the issue of sudden attacks/outliers to the system. In addition, we have also proposed that we can also get a better estimate of the odometry by fusing the odometry data from two different SLAM algorithms to obtain a better odometry estimate of the autonomous system.

Our future work will focus on proposing a solution where we can secure the system if the attacker chooses to inject false data on the sensor's raw measurements. We also plan to extend this work to distributed MCC-EKF security for vehicle to vehicle network in which multi-robot system research [5]–[7], [12], [15], [18]–[21], [25], [27]–[30], [32], [33] can be utilized.

References

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

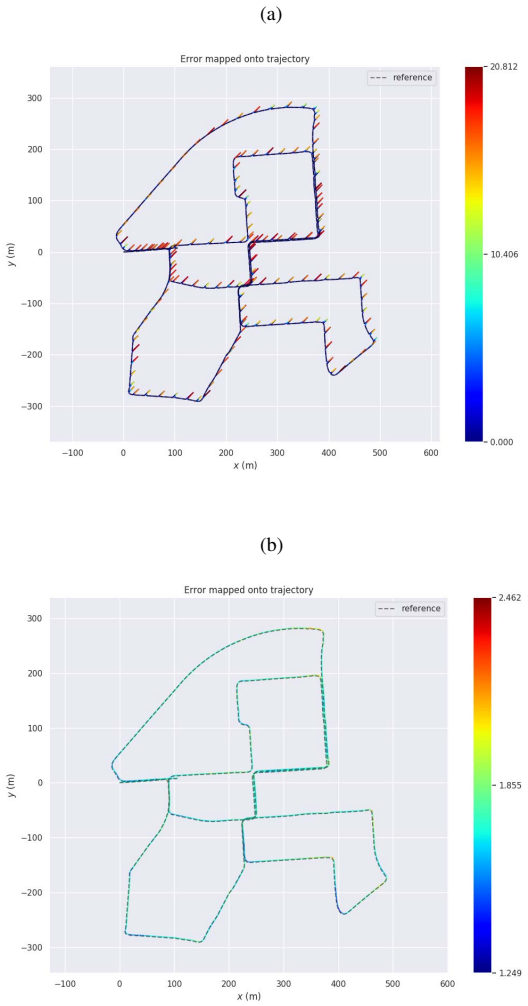


Fig. 8: Evaluation on KITTI sequence number 27. Here, (a) Normal EKF response to attacks in Lidar data, RMSE: 3.193625; (b) MCC-EKF reponse to attacks, RMSE: 0.142083.

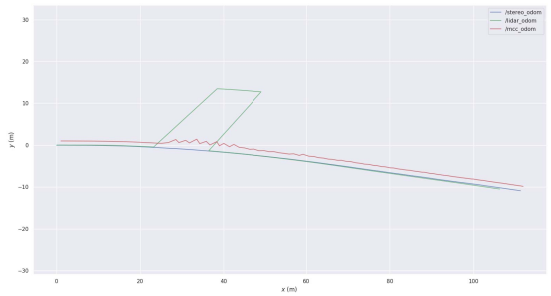


Fig. 9: EKF response on KITTI dataset SEQ 01 when attack vector is introduced.

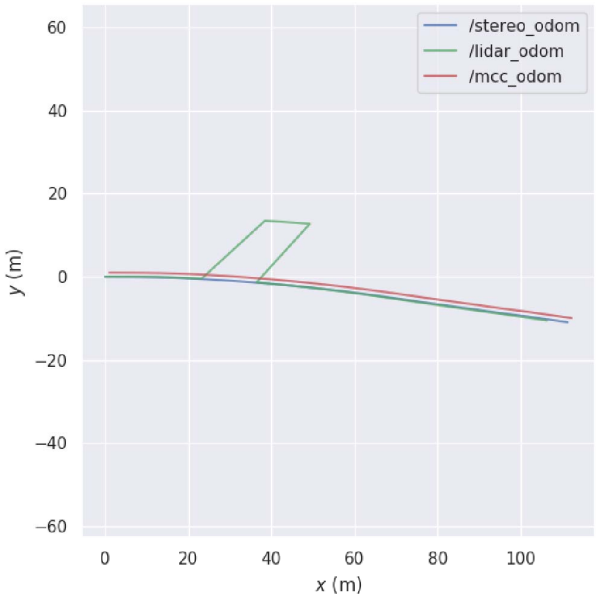


Fig. 10: MCC-EKF response on KITTI dataset SEQ 01 when attack vector is introduced.

[2] D. Brown. Here's how hackers are making your tesla, gm and chrysler less vulnerable to attack. https://www.usatoday.com/story/tech/2019/07/05/tesla-gm-hire-hackers-make-your-connected-car-safer/1594717001/, 2019.

[3] 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.

[4] B. Chen, X. Liu, H. Zhao, and J. C. Principe. Maximum correntropy kalman filter. *Automatica*, 76:70 – 77, 2017.

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

[6] A. D. Dang, H. M. La, and J. Horn. Distributed formation control for autonomous robots following desired shapes in noisy environment. In *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 285–290, Sep. 2016.

[7] A. D. Dang, H. M. La, T. Nguyen, and J. Horn. Formation control for autonomous robots with collision and obstacle avoidance using a rotational and repulsive force based approach. *International Journal of Advanced Robotic Systems*, 16(3):1729881419847897, 2019.

[8] S. Fakoorian, A. Mohammadi, V. Azimi, and D. Simon. Robust Kalman-Type Filter for Non-Gaussian Noise: Performance Analysis With Unknown Noise Covariances. *Journal of Dynamic Systems, Measurement, and Control*, 141(9), 05 2019.

[9] S. Gibb, T. Le, H. M. La, R. Schmid, and T. Berendsen. A multi-functional inspection robot for civil infrastructure evaluation and maintenance. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2672–2677, Sep. 2017.

[10] M. Grupp. evo: Python package for the evaluation of odometry and slam. https://github.com/MichaelGrupp/evo, 2017.

[11] R. Izanloo, S. A. Fakoorian, H. S. Yazdi, and D. Simon. Kalman filtering based on the maximum correntropy criterion in the presence of non-gaussian noise. In *CISS*, pages 500–505. IEEE, 2016.

[12] L. Jin, S. Li, H. M. La, X. Zhang, and B. Hu. Dynamic task allocation in multi-robot coordination for moving target tracking: A distributed approach. *Automatica*, 100:75 – 81, 2019.

[13] H. M. La, N. Gucunski, K. Dana, and S.-H. Kee. Development of an autonomous bridge deck inspection robotic system. *Journal of Field Robotics*, 34(8):1489–1504, 2017.

[14] H. M. La, N. Gucunski, Seong-Hoon Kee, J. Yi, T. Senlet, and Luan Nguyen. Autonomous robotic system for bridge deck data collection and analysis. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1950–1955, Sep. 2014.

[15] H. M. La, R. Lim, and W. Sheng. Multirobot cooperative learning for predator avoidance. *IEEE Transactions on Control Systems Technology*, 23(1):52–63, Jan 2015.

[16] H. M. La, R. S. Lim, B. Basily, N. Gucunski, J. Yi, A. Maher, F. A. Romero, and H. Parvardeh. Autonomous robotic system for high-efficiency non-destructive bridge deck inspection and evaluation. In *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1053–1058, Aug 2013.

[17] H. M. La, R. S. Lim, B. B. Basily, N. Gucunski, J. Yi, A. Maher, F. A. Romero, and H. Parvardeh. Mechatronic systems design for an autonomous robotic system for high-efficiency bridge deck inspection and evaluation. *IEEE/ASME Transactions on Mechatronics*, 18(6):1655–1664, Dec 2013.

[18] H. M. La and W. Sheng. Flocking control of a mobile sensor network to track and observe a moving target. In *2009 IEEE International Conference on Robotics and Automation*, pages 3129–3134, May 2009.

[19] H. M. La and W. Sheng. Flocking control of multiple agents in noisy environments. In *2010 IEEE International Conference on Robotics and Automation*, pages 4964–4969, May 2010.

[20] H. M. La and W. Sheng. Distributed sensor fusion for scalar field mapping using mobile sensor networks. *IEEE Transactions on Cybernetics*, 43(2):766–778, April 2013.

[21] H. M. La, W. Sheng, and J. Chen. Cooperative and active sensing in mobile sensor networks for scalar field mapping. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):1–12, Jan 2015.

[22] M. Labbé and F. Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[23] 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.

[24] 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.

[25] M. T. Nguyen, H. M. La, and K. A. Teague. Collaborative and compressed mobile sensing for data collection in distributed robotic networks. *IEEE Transactions on Control of Network Systems*, 5(4):1729–1740, Dec 2018.

[26] P. Nguyen, H. Nguyen, D. Nguyen, T. N. Dinh, H. M. La, and T. Vu. Parksense: Automatic parking positioning by leveraging in-vehicle magnetic field variation. *IEEE Access*, 5:25021–25033, 2017.

[27] T. Nguyen, T. Han, and H. M. La. Distributed flocking control of mobile robots by bounded feedback. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 563–568, Sep. 2016.

[28] T. Nguyen and H. M. La. Distributed formation control of nonholonomic mobile robots by bounded feedback in the presence of obstacles. In *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 206–211, July 2017.

[29] T. Nguyen, H. M. La, T. D. Le, and M. Jafari. Formation control and obstacle avoidance of multiple rectangular agents with limited communication ranges. *IEEE Transactions on Control of Network Systems*, 4(4):680–691, Dec 2017.

[30] R. Olfati-Saber. Distributed kalman filtering for sensor networks. In *2007 46th IEEE Conference on Decision and Control*, pages 5492–5498, Dec 2007.

[31] OSRF. car_demo. https://github.com/osrf/car_demo, 2019.

[32] H. D. Pham, H. M. La, and T. D. Ngo. Adaptive hierarchical distributed control with cooperative task allocation for robot swarms. In *The 12th IEEE/SICE International Symposium on System Integration (SII), Hawaii, USA.*, pages 1–6, Jan 2020.

[33] H. X. Pham, H. M. La, D. Feil-Seifer, and M. C. Deans. A distributed control framework of multiple unmanned aerial vehicles for dynamic wildfire tracking. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–12, 2018.

[34] A. Sehgal, A. Singandhupe, H. M. La, A. Tavakkoli, and S. J. Louis. Lidar-monocular visual odometry with genetic algorithm for parameter optimization. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, D. Ushizima, S. Chai, S. Sueda, X. Lin, A. Lu, D. Thalmann, C. Wang, and P. Xu, editors, *Advances in Visual Computing*, pages 358–370, Cham, 2019. Springer International Publishing.

[35] A. Singandhupe and H. M. La. A review of slam techniques and security in autonomous driving. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 602–607, Feb 2019.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] E. Yağdereli, C. Gemci, and A. Z. Aktaş. 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 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.