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/geometrical 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, that 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 an open 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, 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 it’s naturally 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

Table I shows the variables and it’s usages throughout the paper.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Usages</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_k</td>
<td>Time instances where k = 0, 1, ..., T</td>
</tr>
<tr>
<td>w_k</td>
<td>System process noise</td>
</tr>
<tr>
<td>v_k</td>
<td>Measurement noise</td>
</tr>
<tr>
<td>F</td>
<td>System Matrix</td>
</tr>
<tr>
<td>H</td>
<td>Observation matrix</td>
</tr>
<tr>
<td>Q_k</td>
<td>Covariance matrix of the system process noise</td>
</tr>
<tr>
<td>R_k</td>
<td>Covariance matrix of the measurement noise</td>
</tr>
<tr>
<td>G_\sigma</td>
<td>Gaussian kernel function with bandwidth ( \sigma )</td>
</tr>
<tr>
<td>w_k</td>
<td>Priori estimate of the state x_k</td>
</tr>
<tr>
<td>J</td>
<td>Objective function</td>
</tr>
<tr>
<td>K_k</td>
<td>Kalman gain</td>
</tr>
<tr>
<td>x_k'</td>
<td>Transpose of a matrix</td>
</tr>
</tbody>
</table>

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:

\[
V_\sigma(X, Y) = E[\kappa_\sigma(X - Y)].
\]  

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

\[
V_\sigma(X, Y) = \frac{1}{N} \sum_{i=1}^{N} G_\sigma(b_i - c_i),
\]  

where \( b_i \) and \( c_i \) are \( N \) random samples drawn from \( X \) and \( Y \) respectively. Equ. (2) plays a key role in our work where, \( G_\sigma(b_i - c_i) = \exp\left(-\frac{|b_i - c_i|^2}{2\sigma^2}\right) \),

and \( \sigma \) is the bandwidth or the kernel size of the Gaussian kernel. From Equ. (3) it becomes evident that if \( b_i = c_i \) Gaussian Correntropy is maximum (this is because if \( b_i = c_i \) then \( \exp(0) = 1 \), 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:

\[
x_k = Fx_{k-1} + w_k,
\]  

\[
y_k = Hx_k + v_k,
\]  

where \( x_k \in \mathbb{R}^n \) is the state vector, \( y_k \in \mathbb{R}^n \) is the measurement vector, \( w_k \) is the system process noise, and \( v_k \) is the measurement noise. Both \( w_k \) and \( v_k \) 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:

\[
J = \frac{1}{2}(y_k - Hx_k)\,\text{tr}^{-1}(y_k - Hw_k) + \frac{1}{2}(x_k - Fw_{k-1})\,\text{tr}^{-1}(x_k - Fw_{k-1}).
\]  

The KF can be derived by solving

\[
\frac{\partial J}{\partial w_k} = 0.
\]  

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

\[
P_{k|k-1} = FP_{k-1|k-1}F' + Q_k,
\]  

\[
K_k = P_{k|k-1}H'(HP_{k|k-1}H' + R_k)^{-1},
\]  

\[
\hat{w}_k = \hat{x}_k - w_k,
\]  

\[
P_{k|k} = (I - K_kH)P_{k|k-1}(I - K_kH)' + K_kR_kK_k',
\]  

where \( K_k \in \mathbb{R}^{n \times m} \) is the Kalman gain, and \( \hat{x}_k \) is the priori estimate of the state \( x_k \). It is based on measurements up to and including time \( k - 1 \), and has covariance \( P_{k|k-1} \). \( \hat{w}_k \) 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
estimating the 6 DOF position of the Lidar sensor. For our
Closest Point (ICP) SLAM, which uses 3D Lidar data for
the overall odometry of the autonomous system (e.g., car).
the two SLAM methods [10], [22] can be used to enhance
We attempt to evaluate how the odometry information from
simulated environment from an open-source repository [31].

Velodyne Lidar (VLP-32) and a simulated stereo camera
different sensors to accomplish our goal. We use a simulated
Figure 1 gives the basic system architecture of our proposed

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

$$
l_k = \frac{G_\sigma(||y_k - H\hat{x}_k||_R^{-1})}{G_\sigma(||\hat{x}_k - F\hat{x}_{k-1}||_{R^{-1}}_{k-1})},$$

$$K_k = (P_{k|k-1}^{-1} + L_kH^RT_k^{-1}H)\hat{x}_k - L_kH\hat{x}_k,$$

$$P_{k|k} = (I - K_kH)P_{k|k-1} - K_kR_kK_k^T.$$


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

$$
\lim_{y_k \to \infty} L_k = 0,
\lim_{L_k \to 0} K_k = 0,
\lim_{K_k \to 0} \hat{x}_k = \hat{x}_k^-.
$$

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 a 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 the odometry information from
the two SLAM methods [10], [22] can be used 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 downscaled for
reducing computational complexity. We use a voxel grid
filter of size 0.2 to downscale 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 then 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 odomet-
try 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. (17) gets
updated as:

$$
l_k = \frac{G_\sigma(||y'_k - H\hat{x}_k||_R^{-1})}{G_\sigma(||\hat{x}_k - F\hat{x}_{k-1}||_{R^{-1}}_{k-1})}.
$$

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
state is updated 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 (Algorithm 1) for our approach is
given below.

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 built-in Lidar and camera
system. The source code and the model description is
Algorithm 1: MCC-EKF algorithm for autonomous system security.

Result: Computed odometry from MCC-EKF SLAM

Lidar Odom (ICP) \( \rightarrow \mathbf{L}_o(x, y, z, r, p, y) \)
Stereo Odom (F2M) \( \rightarrow \mathbf{S}_o(x, y, z, r, p, y) \)

Initialization;
Lidar odom \( \rightarrow (\mathbf{L}_o) \)
Stereo odom \( \rightarrow (\mathbf{S}_o) \)
Compute \( \mathbf{x}_0 \) from Equ. (13)
Compute \( \mathbf{P}_0 \) from Equ. (14)

Prior Estimation;
Compute \( \mathbf{x}_{k|k-1} \) from Equ. (15)
Compute \( \mathbf{P}_{k|k-1} \) from Equ. (16)

while get \( \mathbf{L}_o \) and \( \mathbf{S}_o \):
\[ \text{do} \]
Compute \( \mathbf{L}_k \) from Equ. (24)
Compute Gain \( \mathbf{K}_k \) from Equ. (18)
Update state \( \mathbf{x}_k \) from Equ. (19)
Update \( \mathbf{P}_{k|k} \) from Equ. (20)
\[ \text{end} \]

available as open source [31]. Figure 2 shows the simulated gazebo environment for our work. We have edited the prius car model in order to include a stereo system so that we can get the odometry from the stereo camera (using rtabmap’s FrameToMap (F2M)).

![Fig. 2: Gazebo Simulation Environment.](image)

Figure 3 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 camera. We are injecting attacks on the Lidar odometry and evaluate how the MCC-EKF responds. Generally, for our framework, we can use any two SLAM algorithms to output odometry.

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

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 4. Figure 4(a) shows the Lidar odometry and the map. Figure 4(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 odometry as well as the mapping is shown in Figure 4(b) and Figure 4(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 4 is shown in Figure 5. 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 sim-
We compared the Root Mean Square Error (RMSE) values of individual trajectories with respect to the ground truth. Table II 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 III 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 of the Gaussian kernel function $G_{\sigma}$ plays an important role in MCC-EKF implementation. For our experiment, it is set to 10. Figure 6 shows the response of the MCC-EKF to the attacks on the Lidar data. In Figure 6 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.

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 approach, the attacks are rejected.
We have evaluated the 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 8 and 9. Figure 8 shows the normal EKF response with attacks while Figure 9 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

### 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


