Localization of vehicles using EKF and Traffic Infrastructure Cameras

Aravind Swaminathan, Rajat Sahore

***Abstract*—Currently, autonomous driving is one of the most discussed and researched topic in the automotive industry. In this paper, we have proposed algorithms that can be utilized to accurately detect and localize the autonomous as well as non autonomous vehicles in the simulated environment. We have utilized state of the art methods in robotics such as Extended Kalman Filter and machine learning models such as Yolo v4 detection model i.e. an open source framework of neural networks for object detection. This will help in solving the localization problems for autonomous as well as non autonomous vehicles. We have also proposed and developed a Graphic User Interface (GUI) that can be utilized for measuring, logging and testing the performance of the algorithms utilized in this paper.**

***Index Terms*—ROS, MORSE, MATLAB, autonomous driving, machine learning, robotics, estimation, measurement, perception, graphic user interface**

I. Introduction

Autonomous vehicles are considered as a solution to reduce the road accidents, improve efficiency and convenience for the users etc.[1] The market of autonomous vehicles is expected to be around 58 million vehicles by the year 2030. [2] It is also estimated that 1 out of 10 vehicle will be autonomous by that time. [3]

Localization is one of the fundamental requirements of autonomous driving. It is a process of finding the location of mobile vehicle/robot in the environment. This enables the vehicle to make decisions such as path planning obstacle avoidance etc. [4] Extended Kalman Filter is one of the sensor fusion algorithm that perceive and fuses the information from various sensors such as GPS, IMU, wheel encoders etc. in order to estimate the vehicle's position accurately.[5]

With the introduction of autonomous vehicles, the continuous interaction with the surrounding vehicles has also become more and more significant. This information is particularly important for autonomous vehicles in order to take necessary safe actions that includes planning a safe path for the vehicle. To predict dynamic objects such as the surrounding vehicles, the easiest way is to treat a dynamic obstacle as static. This assumption can help in removing the errors in the existing algorithms that assume that they will move along the current speed and heading. This might not be true in every scenario and it also requires very accurate information regarding their velocity, trajectory etc. This issues in sensors such as unreliable/erroneous sensor data might lead to inefficient/unsafe path planning, jerks in the movement etc.[6]

Infrastructure resources based positioning systems are emerging as one of the solution to track devices that can also help reduce the computation or applications on the device side.[7] Infrastructure Enabled Autonomy (IEA) is a recently proposed model in the field of autonomous driving. This research aims to create a distributed intelligence architecture and includes the transferring of the main capabilities such as sensing and localization to the infrastructure. This model can be utilized for designing scalable systems for autonomous driving applications. [8]

A. Problem Formulation:

The goal of our project is to efficiently localize the position of the vehicles using existing robotics and machine learning algorithms. With the increasing number of autonomous vehicles on the road, it is important to localize the position of the autonomous vehicles accurately so that they can plan a safe path. But still, not all the cars driving on the roads will be autonomous. So, not only the localization of the autonomous vehicles is important but also the position of the other surrounding vehicles is also important for the collision avoidance and path planning algorithms to work efficiently. In some situations, due to physical barriers the environment perception sensors such as LIDAR and camera might not be able to detect an approaching vehicle. Since the surrounding vehicle might be non autonomous, they might also lack localization and V2V/V2X communication technologies. Through V2V/V2X communication, only the autonomous vehicles can share their perceived information from the sensors. Hence, we need a solution for instance, V2X communication that can perceive this information such as vehicle position of the vehicles and provide it on to the autonomous vehicle. This can help in not only detecting other vehicle's position but also provide a feedback for the autonomous vehicle in localization. In our simulation, we tried to create such a scenario with the help of physical barriers and efficiently localize our autonomous and non autonomous vehicles.

II. Adopted Models

A. Communication System

The communication model of our system is Ethernet for all the sensors on the vehicle, and V2X for communication between the autonomous vehicles and infrastructure to the autonomous vehicle. Both, V2X and Ethernet can communicate in a standard way at a very fast rate. Ethernet uses IEEE 802 whereas V2X uses IEEE 802.11p standards for communication.[9] The proposed communication model of our system is shown in Fig 1.

In our implementation, we have used ROS software which was developed keeping distributed computing in mind. [10]

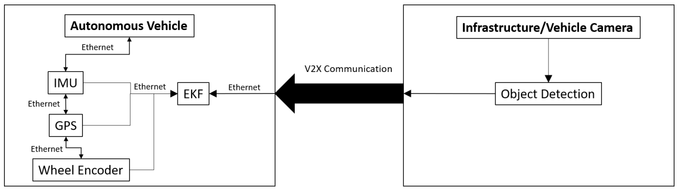

Fig. 1. Proposed Communication System

We have created various nodes and topics to communicate the messages within different nodes. For simplicity, we are running all nodes on the same system but they can also be created on different machines and a network can be setup using TCP/IP.

B. System Model:

Filtering and estimation are two of the most important modules for any autonomous vehicle. In real world applications the state estimation occurs with high sensor noise. Some kind of state estimation algorithm is required to fuse the data from different sensors together to produce an accurate estimate of the system state. In linear systems, the dynamics and measurement models are linear . So the state of the system can be computed using kalman filter. However, in most real-time applications system dynamics and measurement equations are nonlinear. The general state space model of the non-linear system is given below. The system is assumed to be modelled with Gaussian noises.

$$x(k+1) = f_k(x(k), u(k), v(k)) \tag{1}$$

$$z(k) = h_k((x(k), \epsilon(k)) \tag{2}$$

where,
$x(k)$ - represents the state of system
$z(k)$ - represents the measurement from sensors
$v(k)$- Gaussian Process noise $\sim N(0, Q)$
$\epsilon(k)$- Gaussian measurement noise $\sim N(0, R)$
$f_k(.)$- Non linear system dynamic model
$h_k(.)$ - Measurement Model

The system model we used is a 15 dimensional system (including position, orientation, linear velocity, angular velocity and linear acceleration) but, for our results and analysis we are only interested in the position x, position y and yaw states as shown in Eq. 3

$$x_k = [P_x P_y \alpha] \tag{3}$$

C. Sensor Model:

*GPS:*The most important sensor for autonomous cars is GPS for navigation. GPS and GNSS produces output in latitude and longitude in global coordinate system. The output of Position(P_x, P_y, P_z) in (3) is provided by GPS. [11] In our sensor model, we have considered a Gaussian noise due to random effects such as atmospheric conditions, multi path effects etc.

IMU: An IMU is a unique sensor which has no relatable parameters to external world. It operates in its own coordinate system. IMU generally provides data with 6 or 9 Dof with Orientation , linear acceleration and angular acceleration. In our sensor model, we have considered a Gaussian noise due to random effects such as drift effect.

*Wheel Encoders:*A typical sensor which is placed behind every motor. Each wheel encoder is used to count the number of times the motor has rotated. Based on this the velocity of the robot can be computed using the robot dynamic model(differential drive model in this case). [12] In our sensor model, we have considered a Gaussian noise due to random effects such as slipping wheel.

Infrastructure Traffic Camera: In urban environments, with tall buildings and structures on road, the signal of GPS is very poor or even completely erroneous at the ground level.If the major sensor for pose estimation fails, the entire localization module might collapse. So in such complex regions, a traffic camera can be used to obtain the relative position of vehicle and send the information to the car in the global coordinate system. In our sensor model, we have considered a Gaussian noise due to random effects such as vehicle orientation which is discussed in later section.

Vehicle Camera: All the autonomous vehicles are equipped with camera sensor. Again, if the GPS performance is poor, the entire localization module might collapse. So, a camera present in the vehicles can be used to obtain the relative position of vehicle and send the information to the car n the global coordinate system. In our sensor model, we have considered a Gaussian noise due to random effects such as vehicle orientation which is discussed in later section.

D. Simulation Environment and Assumptions:

The traffic camera detection system that we modelled will differentiate the ego vehicle (black) and other vehicles (blue and red). The blue vehicle is also autonomous, we are using the vehicle camera to calculate the position of the ego vehicle (black). As per our research problem, we designed the simulation environment is in such a way that the one vehicle is occluded as shown in the figure 2. This simulation environment is modelled using Morse simulation engine(described in section IV). We have taken the assumption that there is negligible noise/error in the position of the blue vehicle. This position data of the blue vehicle in combination with the object detection from camera sensor is used to calculate the position of the ego vehicle (black)

III. Solution:

In this section, we will describe more about the algorithms and estimators that are used to detect the vehicles. The basic idea of localizing the vehicles from traffic camera and vehicle camera is that it can pass on vehicles location information to the intelligent vehicles. This is an efficient way of environmental understanding for intelligent vehicles which sets

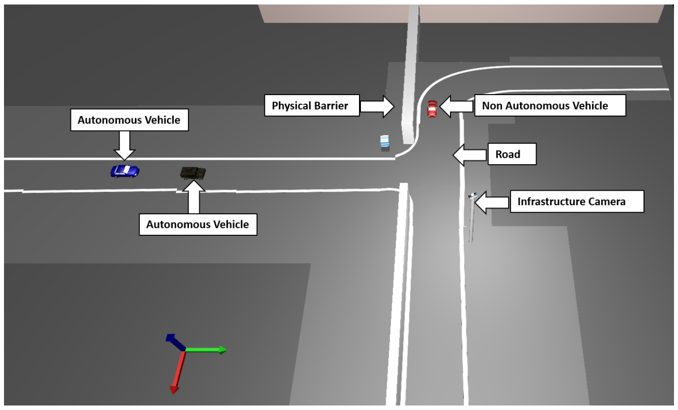

Fig. 2. Top View of the simulation environment

up the base for V2X(Vehicle-to-Everything) based navigation. The traffic camera and vehicle camera were chosen as depth camera, which contains a RGB stream that can be used for detection of bounding box of the vehicles. The depth stream is used to compute the relative position of vehicle from the camera system.

Traffic camera based vehicle positioning using YOLOv4: Object detection and recognition is a general term to describe a collection of related computer vision tasks that involve identifying objects in frames. Thanks to the growth in field of artificial intelligence which help solve these tasks in an easy and efficient way. Some Popular AI models which solve these problems are R-CNN(Region based Convectional Neural Network) and its families followed by YOLO (You Only Look Once) model and its families of network.[13]

In this project, a mini version of yolo-v4(yolov4-tiny) is chosen so that the model is lightweight for the particular task without compromising the accuracy of detection. A typical YOLOv4's architecture is composed of CSPDarknet53 as a backbone, spatial pyramid pooling additional module, PANet path-aggregation neck, and YOLOv3 head [14]. YOLOv4-tiny is the compressed version of YOLOv4 . It is proposed based on YOLOv4 to make the network structure simpler and reduce parameters so that it becomes feasible for developing real time application on embedded devices. YOLOv4-tiny is generally used for much faster training and faster detection. It has only two YOLO heads whereas YOLOv4 has three and the number of pre-trained convolutional layers are 29 when compared to 137 in YOLOv4 which increases the detection time. The Frames Per Second (FPS) in YOLOv4-tiny is approximately eight times that of YOLOv4 [15]. However there is small trade-off with accuracy when compared to with the full network. The reduction in accuracy due to a smaller network is close to one-thirds.

The output of Yolov4 network after detection will be a bounding box in the image with pixels

$$B = x_{min}, x_{max}, y_{min}, y_{max} \tag{4}$$

The center of object is

$$(C_{Bx}, C_{By}) = (\frac{x_{min} + (x_{max} - x_{min})}{2}, \frac{y_{min} + (y_{max} - y_{min})}{2}) \tag{5}$$

With the corresponding depth value in the depth channel of the traffic camera, we can find the relative position of the vehicle using the following transformation

$$X_{pos} = ((C_{Bx} - k(2))d)/k(0) \tag{6}$$

$$Y_{pos} = ((C_{By} - k(5))d)/k(4) \tag{7}$$

where k represents the intrinsic parameters of camera $k \in \{f_x, 0, c_x, 0, f_y, c_y\}$ and d represents the depth at C_{Bx},C_{By}

Extended Kalman Filter: This is the estimation technique to estimate the state matrix in Eq. 3. The estimation method involves a two-step process to determine the states which are prediction and the update steps. The system state is first predicted using the system dynamics in the prediction step and then with the help of measurement the estimated state will be corrected accordingly.[16] Also, during the process of update step, the Kalman gain which is the major factor for final estimated output will be updated every frame in this step. First order Taylor series expansion is used to linearize the non-linearity in the system. The sensor data in real time scenarios will be erroneous due to real time environmental effects.For example, data from IMU will be more error prone due to its drift effect. To reduce the high frequency error in the system, Extended Kalman Filtering techniques are used to estimate the position of the robot accurately. Data from individual sensors will not be enough to estimate the exact state of the system.[17] So, sensor fusion based localization will help us to fuse the information from different sensors to compute the final state of the system. Also, the data from individual sensors will be at a slower rate in real time scenarios, say GPS at 1Hz . But the whole system requires the localization information at faster rate to take the real time decisions. EKF helps achieve this with providing accurate state of the robot at faster rate than the individual sensor input.

Prediction Step: In the prediction step, we predict the state of the system using the system dynamic model.

$$\hat{x}(k+1)^- = f_k(\hat{x}(k), u(k)) \tag{8}$$

$$P(k+1)^- = A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T \tag{9}$$

where,

$$A(k) = (\frac{f_k(x, u, v)}{dx} \mid x = \hat{x}(k), u = u(k), v = 0) \tag{10}$$

$$G(k) = (\frac{f_k(x, u, v)}{dv} \mid x = \hat{x}(k), u = u(k), v = 0) \tag{11}$$

Update Step: In the update step, we use the Kalman gain to compute the final estimate of state taking into account the measurement information from sensors as well.

$$S(k+1) = H(k+1)P(k+1)^- H(k+1)^T + R(k+1) \tag{12}$$

$$W(k+1) = P(k+1)^- H(k+1)^T S(k+1)^{-1} \tag{13}$$

$$\hat{x}(k+1) = \hat{x}(k+1)^- + W(k+1)(z(k+1) - h_{k+1}(\hat{x}(k+1)^-) \tag{14}$$

$$P(k+1) = (I - W(k+1)H(k+1))P(k+1)^- \tag{15}$$

where,

$$H(k+1) = (\frac{h_{k+1}(x, \epsilon)}{dx} \mid x = \hat{x}(k+1)^-, \epsilon = 0) \tag{16}$$

IV. Implementation:

In this section , we will explain about the the simulator, software components and hardware requirements used in this project.

A. *Morse Simulator:*

First, we created the simulation environment with the help of Blender software. Blender is a free open source software that can be used to design 3D computer graphics.[18] Then, we created three ground robots, one to represent autonomous vehicles (blue and black) and the other to represent non autonomous vehicle (red).

In the autonomous vehicle we added sensors such as wheel encoder, IMU and GPS and the data was published on rostopic defined by us. In the non-autonomous vehicle we didn't add any sensors. In order to simulate noise in the sensor data, we added Gaussian noise to the sensors in the position values with standard deviation of 0.08 m and rotation values with standard deviation of 2 degrees. We also added the pose sensor in order to measure the ground truth of both the cars. For the control of robots, we need to generate force to maneuver the robot. This control was achieved with the help of Keyboard Actuator given in Morse. [19] Through Keyboard Actuator, we can define the keys through which we want to move the robot in the desired direction.

For creating the infrastructure camera, we created a stationary ground robot with physical appearance of a traffic pole. Then, the depth camera was placed at the desired location to fetch images.

B. *ROS:*

ROS, an open-source robot operating system. ”ROS is not an operating system in the traditional sense of process management and scheduling; rather, it provides a structured communications layer above the host operating systems of a heterogeneous compute cluster”.[20] It is one of the most efficient open source tools for communication between multiple modules. The ros version used in this project is ROS Melodic. Fig.3 shows the ROS nodes used in our implementation.

C. *Object classification Training:*

To train the YOLO network, Darknet framework is used. Darknet is an open source neural network framework written in C and CUDA. Darknet is mainly used for Object Detection, but have a different architecture, features compared to the other deep learning frameworks. It is faster than many other NN architectures and approaches like FasterRCNN etc.Tensorflow has a broader scope, but Darknet architecture and YOLO is a specialized framework, and they are on top of their game in speed and accuracy. YOLO can run on CPU but you get 500 times more speed on GPU. Since the project involves building a machine vision stack into application, Roboflow will help in accelerating the process. Roboflow helps through all aspects of computer vision, from uploading, annotating, and

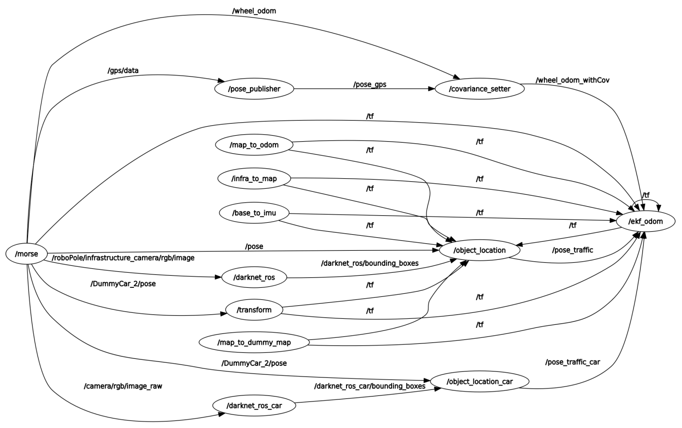

Fig. 3. ROS Nodes of our implementation

organizing your images to training and deploying a computer vision model. This also adds up with the key components of image pre-proessing, image augmentation to the dataset that will be created for model training. The model was trained with a dataset of 122 images with image augmentation. The model was trained for around 4000 iterations. The input image size to the network is 416x416 and augmentation of 13 degrees. After training, the weights are stored locally and later used for prediction using the darknet framework.

D. *Robot Localization- EKF:*

$Robot\_localization$ is a package of state estimation ROS nodes. These ros nodes are implementations to solve nonlinear state estimations for any robots in 2d/3D space. The major features of the $Robot\_localization$ package is fusion of any arbitrary number of sensors, per-sensor input customization, continuous estimation even during a discontinuity in sensor information from any individual sensors. [21] [22] In our implementation, we used EKF algorithm for sensor fusion. Firstly, we converted all the information from vehicle sensors such as IMU, GPS and wheel encoders into the format accepted by $robot_localization$ package. Then, we fused this information and tuned the co-variance matrices to achieve the desired results.

Robot localization takes measurement noise co-variance directly from sensor messages. For the computation of measurement noise co-variance for IMU, GPS and wheel encoders, we used Type A evaluation and computed the variance of the sensor data. For the infrastructure camera, we again did type A evaluation by keeping the position constant and changing the orientation. We found that the measurement was fluctuating around the ground truth with change in orientation. Hence we assumed it to be Gaussian noise as well. We observed negligible offset in X direction and a large offset in the Y direction in the position estimation. This is depicted in Fig. 4 and Fig. 5 for the infrastructure camera. From autonomous car camera, we found similar results. We calculated the mean of ground truth and object detection data in both cases. We then added the difference of the mean values of ground truth and position estimation to our object detection algorithm to remove the offset in the position and also calculated the variance of the position data coming from the object detection algorithm.

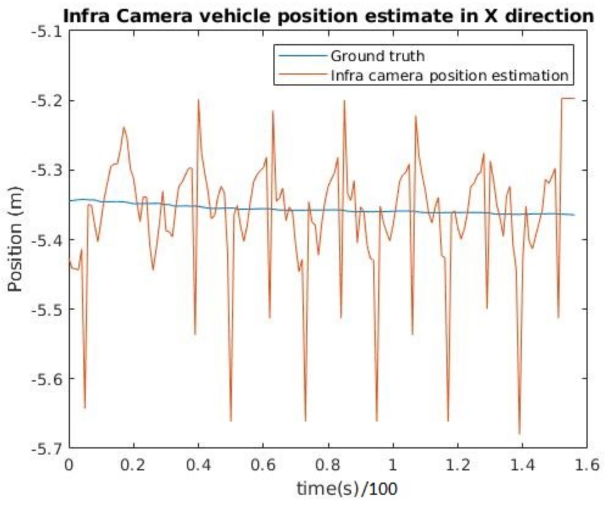

Fig. 4. Variance of Position (m) in X direction using Yolo object detection

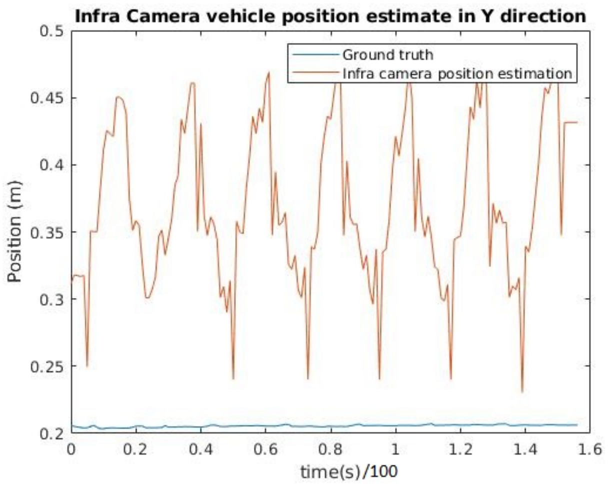

Fig. 5. Variance of Position (m) in Y direction using Yolo object detection

E. Hardware:

The major hardware requirement will be only during the time of training the model . The Hardware used for this current execution of training framework is 16GB of RAM with intel i7v10 processor . Nvidia RTX-3000 Series with 8 GB memory with cuda support to accelerate the training and detection time.

F. MATLAB- GUI:

Firstly, we designed a GUI with the help of MATLAB Apps. [23] The designed GUI is shown in Fig. 6.

In order to read ROS messages, we need to connect to our ROS Network. So, in the GUI we gave a field to enter

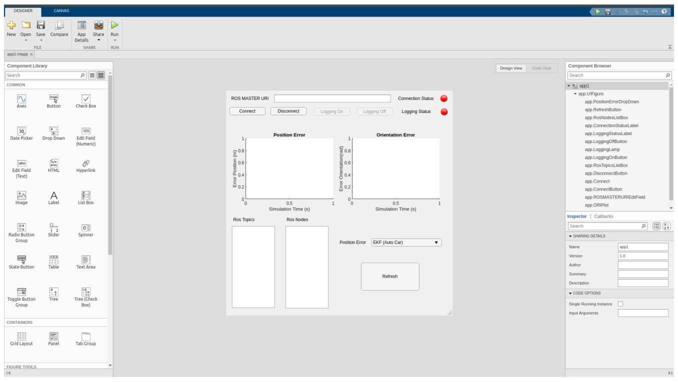

Fig. 6. MATLAB GUI

the ROS_MASTER_URI. Then we gave some buttons to connect to the ROS Network with some status lights as well. Similarly, a button to disconnect from the ROS Network. A subscriber to the ground truth and the data from the camera or the EKF topics was added. This choice between camera and EKF can be selected from a drop down menu.

For EKF data, the error in the position and orientation is calculated based on the data received from the subscribed topics and finally displayed on the graphs. Similarly for camera, only the position error is calculated and displayed on the graph. To check if the rostopics and rosnodes are active, a list was added to check it. Also, we added a refresh button to refresh the active rostopics and rosnodes. Finally, to log the raw messages as well as the calculated results into a mat file, buttons were added to turn logging on and off.

V. Results

This section will present an analysis of the results for each subsystem independently and the combination of the subsystems as mentioned in the sections before. Hence, we will see the results of EKF based localization for autonomous vehicle, Yolov4 detection based localization for autonomous and non autonomous vehicles and finally fusion of Yolov4 detection based localization with EKF for the autonomous vehicle. Throughout this section, the autonomous vehicle is referred as the Ego vehicle (black vehicle) non autonomous vehicle is referred as the Obstacle car (red vehicle) and the other vehicle is referred as another autonomous vehicle(blue car). The vehicle was driven around in the simulation environment and the output of the sub modules were recorded to obtain the results.

A. Yolo v4 Model Results:

In machine vision systems, Average precision is a popular metric in measuring the accuracy of object detection algorithms like SSD, R-CNN. Precision measures how accurate is your predictions. i.e. the percentage of your predictions are correct. The general definition for the Average Precision (AP) is finding the area under the precision-recall curve above. Precision and recall are always between 0 and 1. Therefore, AP falls within 0 and 1 also. Before calculating

AP for the object detection, we often smooth out the zigzag pattern first. mAP (mean average precision) is the average of AP. In some context, we compute the AP for each class and average them. But in some context, they mean the same thing.[24] The graph in Figure 7 represents the mAP(red line). The best mAP that the model could achieve with this data set is 94% and the performance of the model is better after 100 iterations as the average loss is close to 0. The model was trained to detect two classes "Ego" and "Obs" representing ego vehicle and obstacle vehicle. The individual performances of these classes were reported as 87.50% and 98% respectively.

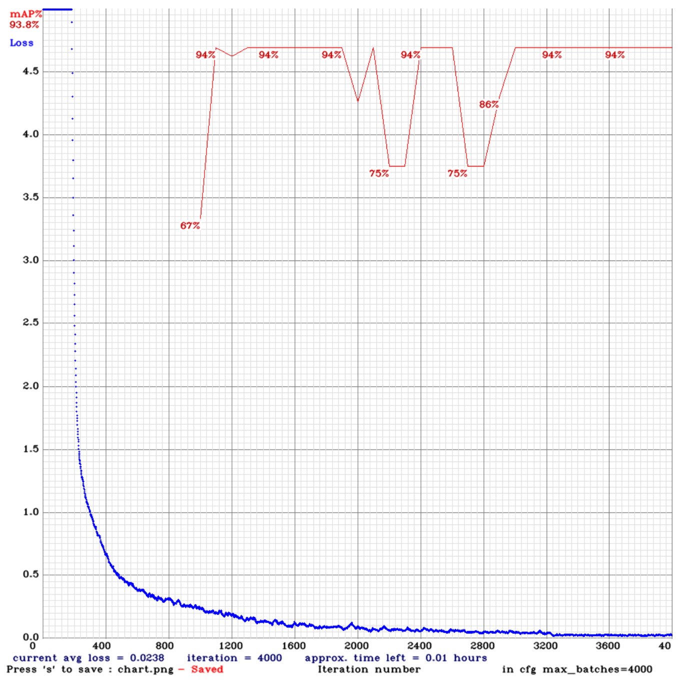

Fig. 7. Model training results

B. EKF:

The position error was calculated based on the position of the ground truth data and EKF results in the X and Y direction. We took two cases, one without tuning the measurement noise and the other without tuning the measurement noise. The summary of the results is depicted in Table I. If we look at the average error in position without tuning the measurement noise co-variance matrix, we were able to localize the car with around 0.1 m accuracy. After tuning the measurement noise co-variance matrix, the position error is reduced and we were able to localize the car with around 0.05 m accuracy. The position error plot is shown in Fig. 8

The orientation error was calculated based on the difference in Euler angle of the ground truth data and EKF results in the Z direction. The summary of the results is depicted in Table II. If we look at the average error in orientation without tuning the measurement noise co-variance matrix, we were able to calculate the car's orientation with very good accuracy with

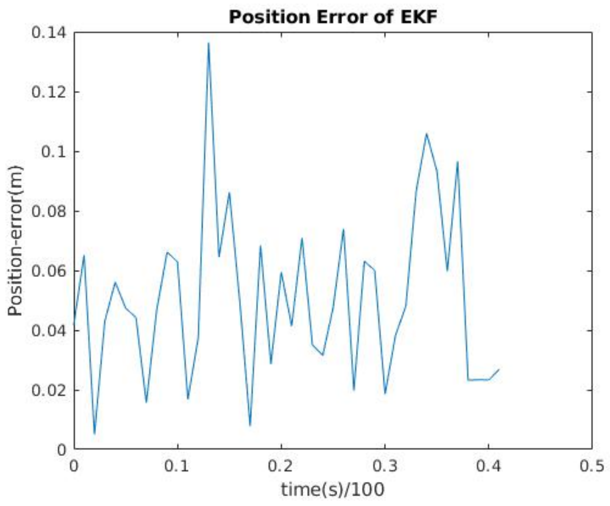

Fig. 8. Ego Car Position error (m) based on EKF with measurement noise tuning

TABLE I
EKF POSITION ERROR SUMMARY

Description	Without Measurement Co-variance Tuning	With Measurement Co-variance Tuning
Max error(m)	0.230	0.1364
Min error(m)	0.018	0.0051
Average error(m)	0.103	0.0509

error value of 2.325e-05 rad. After tuning the measurement noise co-variance matrix, the orientation error was almost same with slight reduction in max error value and we were able to calculate the car's orientation with very good accuracy with error value of 2.192e-05 rad.

TABLE II
EKF ORIENTATION ERROR SUMMARY

Description	Without Measurement Co-variance Tuning	With Measurement Co-variance Tuning
Max error(rad)	0.001	2.3262e-04
Min error(rad)	2.675e-09	2.4232e-07
Average error(rad)	2.325e-05	2.1921e-05

C. Yolov4 Position estimation

The ground truth data for calculation of position error is obtained from the Morse simulator. The accuracy of detection of ego vehicle with traffic camera was around 87.5%. After detection, the depth value is obtained from depth channel and then the corresponding depth value of center pixel is converted to relative position using the Eq. 6 and 7. As we can see from the Fig. 9, the average error for the estimation of ego vehicle position was calculated as 0.1497 m. Similarly, we computed the estimate of ego vehicle from another autonomous car camera. The average error for estimate of ego vehicle based on the other car was 0.09m and can be referred in Fig 10. The average error in obstacle car detection was found to be 0.105m.

TABLE III
EGO CAR POSITION ERROR BASED ON EXTERNAL CAMERAS

Description	**Infrastructure Camera and Ego car**	**Car Camera and Ego car**	**Infrastructure Camera and Obstacle car**
Max error(m)	0.298	0.1779	0.315
Min error(m)	0.035	0.0274	0.068
Average error(m)	0.088	0.0924	0.105

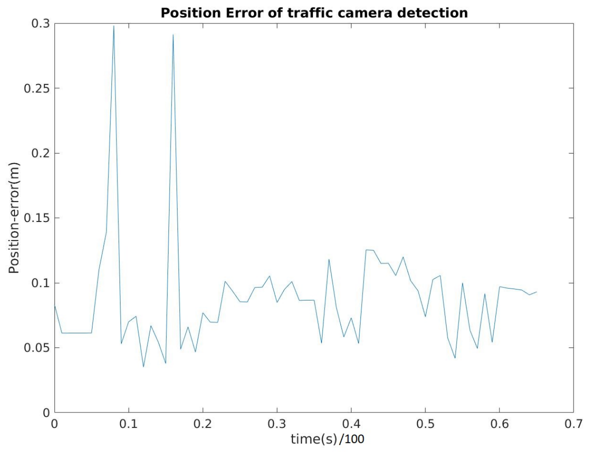

Fig. 9. Ego Car Position error based on Infrastructure camera

D. Integration of EKF with Infrastructure camera and other Ego camera:

Along the input of on-board sensors like GPS, IMU, Wheel odometers, we also fused the data from infrastructure camera and another ego vehicle on the environment. The traffic camera position in the world coordinates is known, so we can apply transformation to convert the position of vehicle from the camera frame to the world frame. Similarly, another autonomous vehicle position is obtained from its pose sensor and data transformation is done from its camera frame to the world frame. This information is passed to ego vehicle as mentioned in communication system block. As we can see in figure 11, the error ranges between 0.011 and 0.263 m with average error of 0.113 m. The low frequency spikes in position estimation by traffic camera were reduced as seen in the graph but the estimation accuracy was almost similar to the only EKF case.

TABLE IV
EGO VEHICLE POSITION ERROR BASED ON EXTERNAL CAMERAS AND ON BOARD SENSORS WITH EKF

Max error(m)	0.256
Min error(m)	0.0125
Average error(m)	0.0632

VI. CONCLUSION

A. Achievements

The aim of this project was to localize the position of autonomous as well as the non autonomous vehicle in the

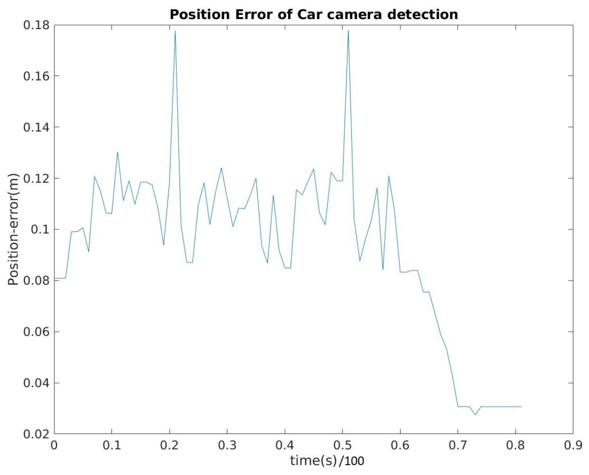

Fig. 10. Ego Car Position error based on another autonomous car camera

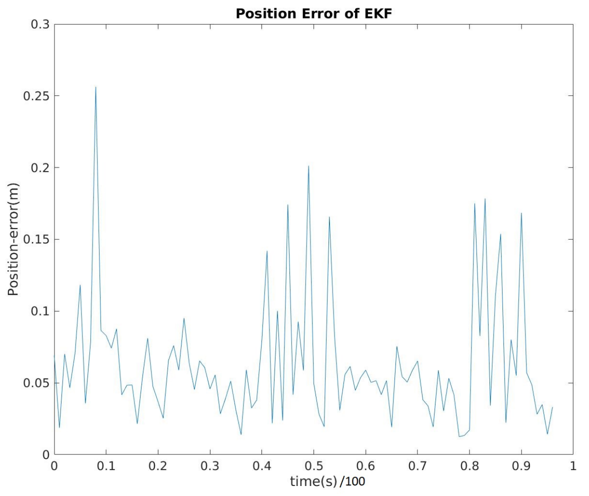

Fig. 11. Ego Car Position error based on external cameras and On board sensors with EKF

simulation environment with the help of vehicle sensors as well as the external resources (in our case camera). This localization objective of autonomous vehicle was achieved with good accuracy by using EKF as the sensor fusion algorithm and tuning the parameters to achieve the desired results.

The object detection using the cameras, also gave satisfactory results to detect the vehicle and predict the vehicle's location. Hence, this method can be utilised to detect the position of vehicles on the road as well as feedback/information fusion for the localization of the autonomous vehicles.

The performance of EKF in the autonomous car after fusing the position data from the infrastructure camera was almost similar with the current sensor noise configuration. If the noise is high then the infrastructure camera based data fusion will be more useful. But, for the detection of non autonomous vehicles the infrastructure camera system is still useful.

The GUI developed in MATLAB helped us in tuning our algorithms as well as analysing the performance of different algorithms used in this project. Finally, our proposed solution predicted locations of vehicles with fair accuracy and can be utilised for collision avoidance/path planning algorithms in order to plan a safe path for autonomous vehicles.

B. Future Works

During the development of this project, we had to put a lot of efforts in tuning the parameters of the Extended Kalman Filter. This process is iterative and consumes a fair amount of time. A possible approach to improve this issue could be to develop an auto tuning solution that can be utilised to tune the EKF in a fast and efficient manner.

For detection using infrastructure resources, we only trained our machine learning algorithm to detect the position of the vehicle. In future, this approach can be extended to detect the vehicle orientation as well. Currently, we tested our performance in simulation with only two autonomous and one autonomous vehicle. It would be interesting to see the performance of this system with many vehicles.

Further, other parameters such as velocity and accelerations of the vehicles can also be detected by adding a radar sensor in the infrastructure resources. Then using camera and radar, we can fuse the information to detect pose as well as the velocity/acceleration data of the vehicle.

References

[1] N. H. T. S. Administration, "Automated vehicles for safety," https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety.

[2] ResearchAndMarkets.com, "Autonomous vehicle market by automation level, by application, by component - global opportunity analysis and industry forecast, 2020-2030," https://www.researchandmarkets.com/reports/5206354.

[3] Statista.com, "06.02.2020 by 2030, one in 10 vehicles will be self-driving globally," https://www.statista.com/press.

[4] S. Huang and G. Dissanayake, *Robot Localization: An Introduction*, 08 2016.

[5] D. Jeon and H. Choi, "Multi-sensor fusion for vehicle localization in real environment," in *2015 15th International Conference on Control, Automation and Systems (ICCAS)*, 2015, pp. 411–415.

[6] D. Ferguson, M. Darms, C. Urmson, and S. Kolski, "Detection, prediction, and avoidance of dynamic obstacles in urban environments," in *2008 IEEE Intelligent Vehicles Symposium*, 2008, pp. 1149–1154.

[7] H. Tran, A. Mukherji, N. Bulusu, S. Pandey, and X. Zhang, "Improving infrastructure-based indoor positioning systems with device motion detection," in *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*, 2019, pp. 176–185.

[8] D. Ravipati, K. Chour, A. Nayak, T. Marr, S. Dey, A. Gautam, S. Rathinam, and G. Swaminathan, "Vision based localization for infrastructure enabled autonomy," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 1638–1643.

[9] V. Mannoni, V. Berg, S. Sesia, and E. Perraud, "A comparison of the v2x communication systems: Its-g5 and c-v2x," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 2019, pp. 1–5.

[10] "Ros documentation," http://wiki.ros.org/Documentation.

[11] Wahyudi, M. S. Listiyana, Sudjadi, and Ngatelan, "Tracking object based on gps and imu sensor," in *2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, 2018, pp. 214–218.

[12] V. Barbulescu, I. Marica, V. Gheorghe, M. Nistor, and M. Patrascu, "Encoder-based path tracking with adaptive cascaded control for a three omni-wheel robot," in *2017 16th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, 2017, pp. 1–6.

[13] S. L. Zicong Jiang, Liquan Zhao and Y. Jia, "Real-time object detection method based on improved yolov4-tiny," 2020.

[14] A. Bochkovskiy and C.-Y. Wang, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[15] Techzizou, "Article on yolov4 vs yolov4-tiny," https://medium.com/analytics-vidhya/yolov4-vs-yolov4-tiny.

[16] G. H. Martin A Skoglund and D. Axehill, "Extended kalman filter modifications based on anoptimization view point," 2015.

[17] L. Wei and Y. Li, "Occluded street objects perception algorithm of intelligent vehicles based on 3d projection model," 2018.

[18] "Blender (software)," https://en.wikipedia.org/wiki/Blender_(software).

[19] "Keyboard actuator," https://www.openrobots.org/morse/doc/1.3/user/actuators/keyboard.html#other-sources-of-examples.

[20] K. S. B. Hasan, "Article - what why and how of ros," https://towardsdatascience.com/what-why-and-how-of-ros-b2f5ea8be0f3.

[21] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," 2018.

[22] E. I. Al Khatib, M. A. Jaradat, M. Abdel-Hafez, and M. Roigari, "Multiple sensor fusion for mobile robot localization and navigation using the extended kalman filter," in *2015 10th International Symposium on Mechatronics and its Applications (ISMA)*, 2015, pp. 1–5.

[23] "Matlab apps," https://de.mathworks.com/discovery/matlab-apps.html.

[24] J. Hui, "Article - map (mean average precision) for object detection," https://jonathan-hui.medium.com/map.