Trajectory Prediction Using Kalman Filter Method As Collision Risk Assessment On Autonomous Tram

1st Fajar Astuti Hermawati
Informatic Department
Universitas 17 Agustus 1945 Surabaya
Surabaya, Indonesia
fajarastuti@untag-sby.ac.id

2nd Nizar Fadila Anhari
Informatic Department
Universitas 17 Agustus 1945 Surabaya
Surabaya, Indonesia
nizarfadila@gmail.com

3rd Bambang Riyanto Trilaksono
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
briyanto@lskk.ee.itb.ac.id

4th Khansa Salsabila Suhaimi
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
23220006@std.stei.itb.ac.id

5th Rini Nur Fatimah
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
rini.rnf@gmail.com

***Abstract*— An autonomous tram is a light rail vehicle (LRV) with its own rail inside the city. This vehicle uses many systems to help its driver maneuver the vehicle. One of those systems is trajectory prediction. Trajectory prediction predicts the collision risk between a tram and another moving object (other vehicle or pedestrian) around it. Kalman filter is one of the most used signal processing algorithms. In this research, the Kalman filter algorithm will be used along with Interacting Multiple Model to predict the trajectory of a moving object around a tram. That information will be used to calculate the time to collision (TTC) and distance to collision (DTC) that will determine the decision tram should take.**

***Keywords*— Autonomous tram, Collision, Kalman filter, Trajectory prediction.**

I. INTRODUCTION

Transportation technology continuously develops to keep up with the era and human needs. One type of vehicle that has not been exempted from this development is the tram. A tram is a light rail vehicle (LRV) that has its own tracks within the city. The tram's rail system can be separate from the roadway, adjacent to the road but separated by supports, or integrated with the roadway.

There is still a lack of awareness and discipline among road users [1]. This is further supported by Indonesia's limited experience operating trams compared to other countries. Implementing trams in Indonesia must be done carefully with safe track designs to reduce the risk of accidents and ensure safety, especially at intersections. Europe reported 7,535 accidents between trams and pedestrians, resulting in 8,802 pedestrian injuries in 2020-2021. Among them, 3% were fatal, 23% were severe, and 74% were minor injuries [2]. Tram drivers are required to always pay attention to the surrounding situation, including pedestrians and other vehicles. Awareness of these situations affects the level of safety and the risk of tram accidents. The high number of accidents caused by trams supports the development of autonomous trams. Autonomous trams are equipped with sensors and decision-making systems to assist drivers in operating trams and increase safety for tram passengers and other road users.

Risk assessment is used to predict the possibility of LRV collisions, where the system would warn the driver if a collision is likely. However, this assistance is limited to warning and stopping the tram if the driver does not respond. The system also needs to address collision avoidance and emergency braking. For such a complex system, the trajectory prediction of other objects on the road needs to be considered [3].

Chen et al. in [4] utilize a hybrid approach combining deep learning and time-varying State-Space Models (SSM), which can be trained end-to-end. This is made possible by a dynamic neural Kalman model that leverages the relative merits of both SSM and deep neural networks. The approach has successfully overcome challenges, such as single-modality estimation in corrupted data, a fusion of multiple sensors in missing data scenarios, and prediction of future trajectories. Using the Data-driven Kalman-based method, [5] performed speed estimation for autonomous racing cars. In the conducted experiments, the performance surpasses that of the Mixed Kalman Filter (MKF), a well-tested algorithm used in autonomous vehicles (such as Pilatus). Trajectory prediction is performed by [6] based on the bimodal extended Kalman filter. However, the research target is limited to pedestrians, and trajectory prediction for vehicles, which will encounter trams on the road, has not been conducted yet. Therefore, in this study, the Kalman filter method is applied. This method is proposed with the hope that the trajectory prediction aligns with the actual predicted object path around the tram, ensuring the safety of both tram users and surrounding objects.

Meanwhile, [7] applies a hybrid method between data-driven and model-based Kalman filter to create KalmanNet. KalmanNet successfully implements the Kalman filter without having prior knowledge about the noise of the input used. Also [8] performed predictions using their custom framework called Prediction with Model-based Planning (PRIME), which, as its name suggests, utilizes a model-based generator to produce trajectories with explicit constraints. It enables accurate multimodal predictions by leveraging a learning-based evaluator to select the most likely trajectory. After comparing it with the state-of-the-art methods at the time, PRIME achieved the lowest miss rate percentage, which was an official scoring metric in the Agroverse 2020 competition. This indicates that PRIME accurately and consistently predicted trajectories in various scenarios.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

Therefore, this study aims to predict the trajectory or path prediction to anticipate the possibility of collisions between autonomous trams and other objects around the tram while it is in motion. Path prediction plays a crucial role in autonomous vehicles as it allows them to effectively observe and understand the behavior of other vehicles around them. With accurate predictions of other vehicles' paths, autonomous vehicles can make appropriate decisions and avoid potentially dangerous situations. The ability to predict paths also enables autonomous vehicles to smoothly adjust their speed and direction of travel while operating in complex and changing traffic conditions. This method is proposed with the hope that trams can avoid collisions with objects around them, such as other vehicles and pedestrians, by predicting the paths they will traverse. Path prediction is a critical component in an autonomous vehicle system that enables vehicles to operate safely, efficiently, and reliably on the road.

II. METHOD

At each time step *t*, the Kalman Filter (KF) estimates x_t based only on the new observation y_t and the previous estimate $\hat{x}_{t-1}$, with a fixed computational complexity [9]. In the Kalman Filter (KF), several matrices are used for estimating and predicting the system state. Here are some commonly used matrices:

1) State Transition Matrix (F): this matrix describes the relationship between the system state at time $\boldsymbol{t}$ and time $\boldsymbol{t}$+1. It determines how the system state evolves.

2) Observation Matrix (H): this matrix describes the relationship between the system state and the observations obtained. It connects the observations to the system state that is being estimated.

3) Covariance Matrix of State Transition Error (Q): this matrix describes the uncertainty or variability in the system state transition over time. Matrix *Q* accounts for the uncertainty in the state transition model.

4) Covariance Matrix of Measurement Error (R): this matrix describes the uncertainty or variability in the obtained measurements. Matrix *R* is used to account for the uncertainty in the observations.

5) Covariance Matrix of Initial State Error (P): this matrix describes the uncertainty or variability in the initial state. Matrix *P* is used to account for the uncertainty in the initial state estimation.

6) Kalman Gain Matrix (K): this matrix combines information from the prediction and actual observations. Matrix *K* describes the extent to which the actual observations influence the system state estimation.

In this study, three types of Kalman Filter (KF) models are used: Constant Velocity (CV), Constant Acceleration (CA), and Constant Turn Rate (CT). The differences between these models lie in each model's F and H matrices. Here are the F and H matrices for each model [9]:

1. Kalman Filter Constant Velocity (KFCV): Δt here at (1) is a time unit in which we use 0.1 second as its value.

a. F Matrix

$$\begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

b. H Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

2. Kalman Filter Constant Acceleration (KFCA): Same as in (1), Δt in (3) is a time unit set as 0.1 second

a. F Matrix

$$\begin{bmatrix} 1 & \Delta t & 0.5 * \Delta t^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0.5 * \Delta t^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

b. H Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

3. Kalman Filter Constant Turn Rate (KFCT): In (5) $\Delta\theta$ is the result of $\Delta\theta * \Delta t$ in which $\Delta\theta$ is equals to π / 180 * *x*. *x* is a hyperparameter with a default value set as 0.1.

a. F Matrix

$$\begin{bmatrix} 1 & \sin(\theta)/\Delta\theta & 0 & -(1-\cos(\theta))/\Delta\theta & 0 \\ 0 & \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & (1-\cos(\theta))/\Delta\theta & 1 & \sin(\theta)/\Delta\theta & 0 \\ 0 & \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

b. H Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

In general, the state equations used in the Kalman Filter can be represented by the following equations:

$$\boldsymbol{x}_{t+1} = \mathbf{F}_t \,.\, \boldsymbol{x}_t + \boldsymbol{E}_t + \boldsymbol{w}_t \quad (7)$$

$$\mathbf{y}_t = \mathbf{H}_t \,.\, \boldsymbol{x}_t + \mathbf{V}_t \quad (8)$$

$\boldsymbol{x_{t+1}}$ in (7) represents the predicted state at time *t*+1, F is the state transition matrix, x_t is the state at time t, $\boldsymbol{E}_t$ is the control input matrix (if applicable), $\boldsymbol{w}_t$is noise progress. $\boldsymbol{y}_t$ in (8) represents the expected measurement at time t calculated using *H*, the measurement matrix multiplied by the current

state at time *t*, while $\boldsymbol{V}_t$ is the measurement noise. There are two steps in Kalman Filter: predicting and updating [9]. Predict step in Kalman Filter represented by the following equations:

$$\widehat{\boldsymbol{x}}_{t|t-1} = \mathbf{F}_{\boldsymbol{t-1}} \cdot \hat{x}_{t-1|t-1} + \boldsymbol{E}_{t-1} \tag{9}$$

$$\mathbf{P}_{t|t-1} = \boldsymbol{F}_{\boldsymbol{t-1}} \cdot \mathbf{P}_{t-1|t-1} \cdot \mathbf{F}^{\mathrm{T}}{}_{\boldsymbol{t-1}} + \boldsymbol{Q}_{\boldsymbol{t-1}} \tag{10}$$

The update step involves utilizing the predicted (a priori) state estimate, denoted as $\widehat{\boldsymbol{x}}_{t|t-1}$, and the expected error covariance, represented as $\mathbf{P}_{t|t-1}$ [9]. The following is the update step of KF:

$$\tilde{\mathbf{y}}_t = \mathbf{y}_t - \mathbf{H}_t \cdot \widehat{\boldsymbol{x}}_{t|t-1} \tag{11}$$

$$\mathbf{S}_t = \mathbf{H}_t \cdot \mathbf{P}_{t|t-1} \cdot \mathrm{H}_t^T + \mathbf{R}_{\boldsymbol{t}} \tag{12}$$

$$x_{t|t} = \boldsymbol{x}_{t|t-1} + \mathbf{K}_t \cdot \widetilde{\mathbf{y}}_t \tag{13}$$

$$\boldsymbol{P}_{t|t} = (1 - \boldsymbol{K}_t \cdot \boldsymbol{H}_{\boldsymbol{t}}) \cdot \boldsymbol{P}_{t-1} \tag{14}$$

$$\mathbf{K}_t = \boldsymbol{P}_{t|t-1} \cdot \boldsymbol{H}_t^T \cdot \mathbf{S}_{\mathrm{t}}^{-1} \tag{15}$$

Using the innovation residual $\tilde{\mathbf{y}}_t$ and its covariance $\mathbf{S}_t$, along with the Kalman gain $\mathbf{K}_t$, the (a posteriori) state estimate $\hat{\mathbf{x}}_{t|t}$ and the estimate covariance $\boldsymbol{P}_{t|t}$ are determined.

The Interacting Multiple Model Kalman Filter (IMM-KF) algorithm consists of four steps in its implementation, namely (1) interaction step, (2) filtering step, (3) model probability update step, and (4) combination step [4]. The state equations in the IMM interaction step are presented as follows.

$$\boldsymbol{x}_{t+1} = \boldsymbol{F}_t^{(i)} + \boldsymbol{E}_t^{(i)} + \boldsymbol{w}_t^{(i)} \tag{16}$$

$$\mathbf{y}_t = \boldsymbol{H}_t^{(i)} \cdot \boldsymbol{x}_t + \boldsymbol{V}_t^{(i)} \tag{17}$$

where (i) denotes the currently active model m_k from the model set $M = m^{(1)}, m^{(2)}, m^{(3)}$ [4]. Transition probability from model $m^{(i)}$ to model $m^{(j)}$ is as follows.

$$\boldsymbol{Pr}(\boldsymbol{m}_t = \boldsymbol{m}^{(i)}) = \boldsymbol{\pi}_{ij} \tag{18}$$

$\boldsymbol{m}_t \in \mathrm{M}$ is the current active model at t time and transition probability denoted as $\boldsymbol{\pi}_{ij}$ with a value between 0 and 1. The basic principle used in IMM-KF is the separate filtering of each model $\boldsymbol{m}^{(i)}$, which is used in parallel, and the estimation results are used to estimate the probabilities of each active $\boldsymbol{m}^{(i)}$. The individual filter estimates in the interaction stage are combined to initialize each filter.

$$\boldsymbol{c}^{(i)} = \sum_{j=1}^{M} \boldsymbol{\pi} \cdot \boldsymbol{j} \cdot \boldsymbol{i} \cdot \boldsymbol{\mu}_{t-1}^{(j)} \tag{19}$$

$$\boldsymbol{\mu}_{t-1|t-1}^{(j|i)} = \frac{\pi_{ji} \cdot \mu_{t-1}^{(j)}}{c^{(i)}} \tag{20}$$

$$\boldsymbol{x}_{k-1|t-1}^{-(i)} = \sum_{j=1}^{M} \boldsymbol{\mu}_{t-1|t-1}^{(j|i)} \cdot \widehat{\boldsymbol{x}}_{t-1|t-1}^{-(j)} \tag{21}$$

$$\boldsymbol{P}_{t-1|t-1}^{-(i)} = \sum_{j=1}^{M} \boldsymbol{\mu}_{t-1|t-1}^{(j|i)} \cdot (\boldsymbol{P}_{t-1|t-1}^{-(j)} + \boldsymbol{X}_{t-1|t-1}^{-(i,j)} \tag{22}$$

using $\boldsymbol{X}_{t|t}^{-(i,j)} = \left(\boldsymbol{x}_{t|t}^{-(i)} - \boldsymbol{x}_{t|t}^{-(j)}\right)\left(\boldsymbol{x}_{t|t}^{-(i)} - \boldsymbol{x}_{t|t}^{-(j)}\right)^T$, the conditional model probability $\boldsymbol{\mu}_{t-1|t-1}^{(j|i)}$ of transitioning from $\boldsymbol{m}^{(j)}$ to $\boldsymbol{m}^{(i)}$, the estimated state of each filter $\widehat{\boldsymbol{x}}_{t-1|t-1}^{-(j)}$, its covariance $\boldsymbol{P}_{t-1|t-1}^{-(i)}$, the mixing of the state estimates $\boldsymbol{x}_{t-1|t-1}^{-(i)}$, and its covariance $\boldsymbol{P}_{t-1|t-1}^{-(j)}$ [9]. In the filtering step, each filter M is executed separately as shown in (16) and (17) to obtain the innovation residual $\widetilde{\boldsymbol{y}}_t^{(i)}$ and covariance $\boldsymbol{S}_t^{(i)}$, as well as the state estimate $\widehat{\boldsymbol{x}}_{t|t}^{(i)}$ and covariance $\boldsymbol{P}_{t|t}^{(i)}$. The KF models for Constant Velocity, Constant Acceleration, and Constant Turn Rate are represented by equations (1), (3), and (5) respectively. In the probability update stage, the innovation residual from each model is used to update the model probabilities [9].

$$\boldsymbol{L}_t^{(i)} = \frac{\exp\left(-\frac{1}{2} \cdot \widetilde{\boldsymbol{y}}_t^{(i)^T} \cdot \boldsymbol{S}_t^{(i)^{-1}} \cdot \widetilde{\boldsymbol{y}}_t^{(j)}\right)}{\left|2\pi \cdot \boldsymbol{S}_t^{(j)}\right|^{1/2}} \tag{23}$$

$$\boldsymbol{\mu}_k^{(i)} = \frac{\boldsymbol{c}^{(i)} \cdot L_t^{(i)}}{\sum_{j=1}^{M} \boldsymbol{c}^{(i)} \cdot L_t^{(i)}} \tag{24}$$

where $\boldsymbol{L}_t^{(i)}$ represents the likelihood of the observation when considering $\widetilde{\boldsymbol{y}}_t^{(i)}$ and the updated model probability $\boldsymbol{\mu}_k^{(i)}$. During the combination step, each model's state estimates and covariances are mixed and weighted by the updated model probabilities. This process ensures that the final estimate incorporates information from all the models based on their respective probabilities.

$$\widehat{\boldsymbol{x}}_{t|t} = \sum_{i=1}^{M} \boldsymbol{\mu}_t^{(i)} \cdot \widehat{\boldsymbol{x}}_{t|t}^{(i)} \tag{25}$$

$$\boldsymbol{P}_{t|t} = \sum_{i=1}^{M} \boldsymbol{\mu}_t^{(i)} \cdot \left(\boldsymbol{P}_{t|t}^{(i)} + \left(\widehat{\boldsymbol{x}}_{t|t} - \widehat{\boldsymbol{x}}_{t|t}^{(i)}\right)\left(\widehat{\boldsymbol{x}}_{t|t} - \widehat{\boldsymbol{x}}_{t|t}^{(i)}\right)^T\right) \tag{26}$$

TTC (Time To Collision) is a time-based measure used to assess the safety level between objects [10]. If no evasive action is taken, it is the remaining time before two or more objects collide. This index is crucial in designing collision avoidance systems [11] and considering when and how drivers should adjust speed. TTC can be calculated using the following formula [12]:

$$TTC_t(t) = \frac{X_o(t) - X_t(t) - l_{\mathrm{o}}}{\dot{X}_{\mathrm{t}}(t) - \dot{X}_{\mathrm{o}}(t)} \tag{27}$$

$$DTC_t(t) = TTC_t \cdot \dot{X}_{\mathrm{t}} \tag{28}$$

where *t* and *o* denote ego vehicle and object, respectively. X marks the location while $\dot{X}$ refers to the speed and l_{o} is the length of the object. A threshold value typically determines the time threshold for acting in collision avoidance situations. This threshold value is used to differentiate between safe and unsafe conditions. The threshold value of TTC depends on

driver behavior, so no exact value distinguishes between safe and hazardous situations. Using a threshold of 4 or 5 seconds often leads to a higher occurrence of false alarms than a threshold of 3 seconds [13]. Drivers without driver assistance systems typically exhibit a minimum threshold of 3-5 seconds, while drivers with driver assistance systems have a threshold of 2.6 seconds [14].

III. Result and Discussion

A. *Testing Scenario*

The trajectory prediction system will be tested in the Carla simulator. A fire truck vehicle will represent the tram as the ego vehicle. The vehicle will drive on a single lane to simulate the tram running from one station to another. TTC (Time to Collision), DTC (Distance to Collision), waypoints, the location of other objects relative to the tram, and object trajectory predictions will be displayed on a GUI plot application. The testing aims to verify if the displayed trajectory predictions align with expectations and if the data is passed to the safety assessment system. The testing will be conducted on one of the following two maps: Town02 (Fig. 1) or Town10 (Fig. 2).

Fig.1, Carla map Town02

Fig.2. Carla map Town10

The testing will be conducted 50 times, using 50 vehicles and 32 pedestrians as objects. The test is successful if the tram (represented by a fire truck sprite) successfully travels from the starting point to the endpoint without colliding with any other objects. Testing can also be performed by keeping the tram stationary and introducing a moving vehicle around it to serve as input for trajectory prediction. Alternatively, an object can be stationary while the tram moves towards it to test the track's object detection sensors. These two methods can serve as alternatives for developing and testing specific algorithms, especially for evaluating small changes made to the system.

B. *KFCV Stress Test*

A stress test was conducted using the Carla simulation software, utilizing one of the default maps, Town 2, as shown in Fig.1 . A run is considered successful if the tram starts from point 1 in Figure and travels to point 2 along the path indicated in Fig.3 without experiencing any collisions or contact with other objects (vehicles and pedestrians).

Out of the 50 test runs performed, 80% were successful, while the remaining 20% failed for various reasons. One of the reasons was a problem with the filter-detecting vehicles on the tram tracks. During the test, the filter could only detect vehicles on the tracks facing the same direction as the tram. Therefore, the next step is to improve the object detection filter for vehicles on the tram tracks.

Fig.3, KFCV stress test route from “1” to “2”

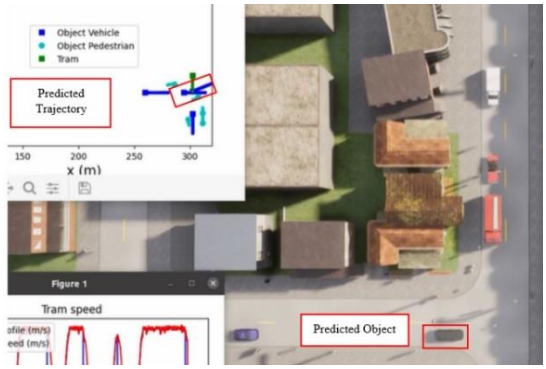

Fig.4, Predicted object trajectory using Kalman Filter Constant Velocity (KFCV) considered not good enough on turning road

C. KFCV Stress Test (Post Filter Fix)

The stress test was conducted following the same rules as before. This stage resulted in a success rate of 84%, a 4% improvement from the previous testing. However, the obtained trajectory predictions were still considered insufficient and too linear during turns (Fig. 4). Therefore, it was decided to switch to the IMM Kalman filter algorithm for trajectory prediction.

D. IMM-KF Stress Test

IMM (Interacting Multiple Model) successfully predicts object trajectories much better than KFCV, as seen in Fig.5. For the 50 test runs, the testing rules and environment remained the same as the previous KFCV algorithm testing. After conducting 50 total test runs, a success rate of 84% was achieved. 90% of collisions were due to imperfect detection of objects on the tram tracks, as shown in the figure. The sensors use object coordinates, which represent the object's center point, to determine whether the object is on the tracks. However, the sensors fail because only a portion of the object is on the tracks, while the center point or object coordinates are still outside. Therefore, the decision-making component will attempt to address this issue. Meanwhile, the IMM algorithm will go through a hyperparameter tuning phase to maximize the results obtained.

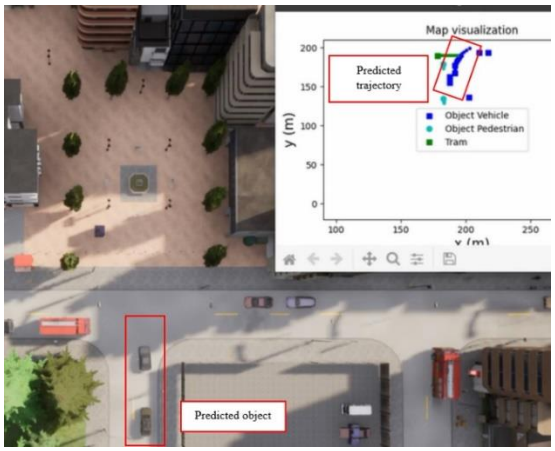

Fig.5, Predicted trajectory using Interacting Multiple Model Kalman Filter (IMM-KF)

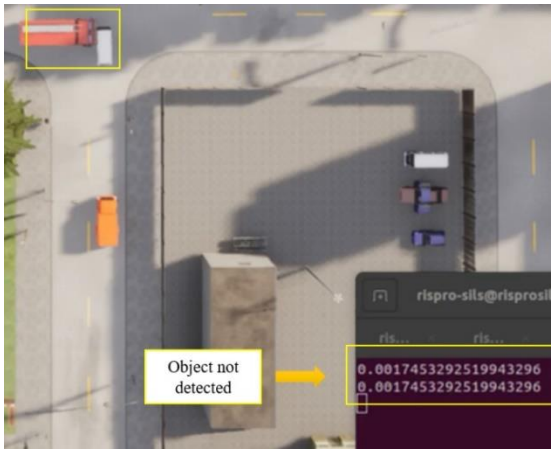

Fig.6, Object on the rail not detected by the sensor.

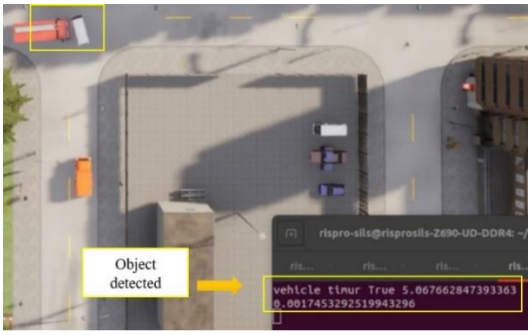

Fig.7, Object on the rail detected.

E. IMM-KF Tuning

To find the best results from the IMM algorithm, tuning is performed on its hyperparameters. The hyperparameters being tuned are the weight variables of the transition matrix and the value of θ˙ (theta). The initial settings before tuning, which were used in the previous stress test, are as follows:

$$P = \begin{bmatrix} 0{,}98 & 0{,}01 & 0{,}01 \\ 0{,}01 & 0{,}98 & 0{,}01 \\ 0{,}01 & 0{,}01 & 0{,}98 \end{bmatrix} \quad (28)$$

With $\dot{\theta}$ = 0,1 radian, max iteration = 100 and max predicted range = 20 meter.

In the transition matrix P (11), the first row contains the CV, CA, and CT weights, respectively. The second row contains CT, CV, and CA weights, and the third row contains CA, CT, and CV weights. These weights determine the influence of each model's output value. The value of θ˙ determines the extent to which the vehicle's inclination affects the CT prediction, indirectly influencing the final prediction of IMM for the turning vehicle. The predicted trajectory distance tends to be directly proportional to the maximum iteration, meaning that a higher maximum iteration will result in a longer predicted trajectory. However, the trajectory length will not exceed the maximum distance limit. After the tuning process, the suitable hyperparameters obtained are as follows:

$$P = \begin{bmatrix} 0{,}9 & 0{,}075 & 0{,}025 \\ 0{,}025 & 0{,}9 & 0{,}075 \\ 0{,}075 & 0{,}025 & 0{,}9 \end{bmatrix} \quad (12)$$

With $\dot{\theta}$ = 1.5 radian, max iteration = 300 and max predicted range = 20 meter.

By reducing the weight of CV and increasing the weights of CA and CT, the algorithm can predict trajectories much better in turning roads. Additionally, by increasing the value of θ˙ to 1.5 radians from the previous 0.1 radians, the algorithm can predict far-reaching turning trajectories even before the object approaches the turn. This can be observed in Fig. 8 and Fig. 9, where in Fig. 8, accurate predictions occur just before the object turns, while in Fig. 9, correct predictions can occur even when the object is still relatively moving straight. Increasing the maximum number of iterations also extends the predicted trajectory, allowing the tram to make decisions more quickly. In the previous maximum of 100 iterations, the predicted distance often did not reach the maximum distance, so the maximum iteration

value was increased during the tuning process. It can also be seen in Fig. 9 that the resulting predicted trajectory is longer compared to the IMM algorithm's results before the tuning process.

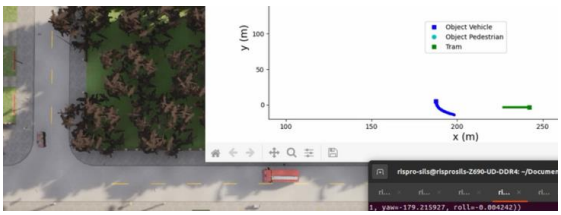

Fig.8, IMM-KF prediction result before tuning process

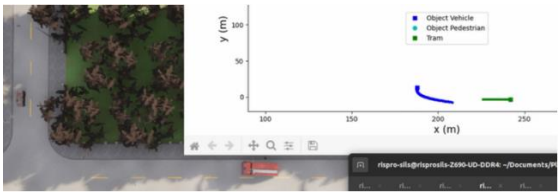

Fig.8, IMM-KF prediction result after tuning process

F. IMM-KF Stress Test (Post Tuning)

This time, the testing was conducted on a straight path (Fig.10) but passed several intersections. The testing used the algorithm that has undergone hyperparameter tuning and the new Finite State Machine design for the decision-making component. It achieved a success rate of 92%, an increase of 8% compared to the testing before the IMM algorithm underwent hyperparameter tuning. 3 out of 4 failures occurred due to inaccurate timing of TTC and DTC communicated to the decision-making component. This could be due to poor prediction results or a loss in the risk assessment component to calculate DTC and TTC accurately.

Fig.10, IMM-KF stress test post tuning route

IV. Conclusion

The IMM-KF algorithm has proven successful as an autonomous tram trajectory prediction algorithm, with a success rate of 92% in a total of 50 test runs. This demonstrates its potential for application in autonomous trams, with room for further development. From the latest test results, there is still an 8% error margin that can be minimized through further refinement. Trajectory prediction plays a significant role in decision-making to avoid and maintain distance from other objects on the road, including but not limited to cars and pedestrians. There is a need for more discrete testing methods to compare the prediction results with the actual state that occurs. This is because the number of predicted coordinates can be many times greater than the number of existing data coordinates.

Acknowledgment

This research work was supported in part by Universitas 17 Agustus 1945 Surabaya under the Research Program in the Institutional Support System Competition Program – Independent Campus (PK-KM) and in part by Institut Teknologi Bandung under the Autonomous Tram Research.

References

[1] M. Damayanti, S. Malkhamah, and K. Walker, "Tramway Management System In Indonesia," *Journal of the Civil Engineering Forum*, vol. 1, no. 1, 2015.

[2] C. Lackner *et al.*, "Tram to Pedestrian Collisions—Priorities and Potentials," *Frontiers in Future Transportation*, vol. 3, Jun. 2022, doi: 10.3389/ffutr.2022.913887.

[3] M. Lüy, E. Çam, F. Ulamiş, I. Uzun, and S. I. Akin, "Initial results of testing a multilayer laser scanner in a collision avoidance system for Light Rail Vehicles," *Applied Sciences (Switzerland)*, vol. 8, no. 4, Mar. 2018, doi: 10.3390/app8040475.

[4] C. Chen, C. X. Lu, B. Wang, N. Trigoni, and A. Markham, "DynaNet: Neural Kalman Dynamical Model for Motion Estimation and Prediction," *IEEE Trans Neural Netw Learn Syst*, vol. 32, no. 12, pp. 5479–5491, Aug. 2019, [Online]. Available: http://arxiv.org/abs/1908.03918

[5] A. L. Escoriza, G. Revach, N. Shlezinger, and R. J. G. van Sloun, "Data-Driven Kalman-Based Velocity Estimation for Autonomous Racing," in *2021 IEEE International Conference on Autonomous Systems (ICAS)*, IEEE, Aug. 2021, pp. 1–5. doi: 10.1109/ICAS49788.2021.9551175.

[6] C. Y. Lin, L. J. Kau, and C. Y. Chan, "Bimodal Extended Kalman Filter-Based Pedestrian Trajectory Prediction," *Sensors*, vol. 22, no. 21, Nov. 2022, doi: 10.3390/s22218231.

[7] G. Revach, N. Shlezinger, R. J. G. V. Sloun, and Y. C. Eldar, "Kalmannet: Data-Driven Kalman Filtering," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 3905–3909. doi: 10.1109/ICASSP39728.2021.9413750.

[8] H. Song, D. Luan, W. Ding, M. Y. Wang, and Q. Chen, "Learning to Predict Vehicle Trajectories with Model-based Planning," Mar. 2021, [Online]. Available: http://arxiv.org/abs/2103.04027

[9] V. Lefkopoulos, M. Menner, A. Domahidi, and M. N. Zeilinger, "Interaction-Aware Motion Prediction for Autonomous Driving: A Multiple Model Kalman Filtering Scheme," *IEEE Robot Autom Lett*, vol. 6, no. 1, pp. 80–87, Jan. 2021, doi: 10.1109/LRA.2020.3032079.

[10] J. C. Hayward, "Near-Miss Determination Through Use Of A Scale of Danger," in *51st Annual Meeting of the Highway Research Board*, Columbia, 1972, pp. 24–34.

[11] R. Van Der Horst and J. Hogema, "Time-To-Collision And Collision Avoidance Systems," in *6th ICTCT Workshop Salzburg*, 1994. [Online]. Available: https://www.researchgate.net/publication/237807114

[12] M. Saffarzadeh, N. Nadimi, S. Naseralavi, and A. R. Mamdoohi, "A general formulation for time-to-collision safety indicator," *Proceedings of the Institution of Civil Engineers: Transport*, vol. 166, no. 5, pp. 294–304, Oct. 2013, doi: 10.1680/tran.11.00031.

[13] S. Hirst and R. Graham, "The Format and Presentation of Collision Warnings," in *Ergonomics and Safety of Intelligent Driver Interfaces*, Y. I. Noy, Ed., CRC Press , 1997.

[14] J. H. Hogema and W. H. Janssen, "Effects of Intelligent Cruise Control on driving behaviour: a simulator study," Rotterdam, Netherlands, 1996.