Robotic Rollator Velocity Estimation

Raiyan Faruqui, Kevin Qiu and Abdullah Yeaser
Department of Mechanical and Mechatronics Engineering
University of Waterloo
Waterloo, Canada
rfaruqui@uwaterloo.ca, longlai.qiu@uwaterloo.ca, aryeaser@uwaterloo.ca

***Abstract*—Robotic rollator development projects have been on the rise as the demand for smarter assistive devices have been increasing with the aging population. Traditional rollators have led to many slips and falls, which can be prevented through the use of robotic systems. An integral component in the development of robotic rollator control systems is the estimation and control of its velocity state. Velocity estimation can be a challenging task, especially for lateral velocity due to unattainable sensor measurement. In this paper, we compare different methods for lateral velocity estimation of robots and propose a Temporal Convolution Network augmented with Kalman Filters for optimal performance. The developed algorithm is able to outperform all other methods for our data set.**

I. Introduction

Mobility is a critical aspect in the daily lives of many individuals. While often overlooked, being mobile can have several beneficial implications on quality of life, such as keeping joints healthy and prevent risk of injury. Mobility impairments are a major risk amongst the elderly, and there now exists increasing demand for assistive devices as the population ages [1], [2]. Studies have shown that lack of mobility has resulted in deterioration of one's health and in severe cases, may lead to increased risk of mortality [1]. Fortunately, assistive devices such as wheelchairs, walkers and canes exist to aid individuals suffering from mobility impairments. While wheelchairs have been prolongedly used to facilitate mobility, the device is limited to certain terrains, not to mention that an extended period seated is detrimental to one's health. Wheelchairs are also much more expensive and is recommended for individuals who have little to no balance and stability control. In contrast, canes provide greater accessibility for users and promotes lower limb activity, however its lack of support can induce greater risk of tripping. Walkers address the pitfalls outlined from both devices and provides an overall safer and healthier alternative.

There are two main types of walkers in the market. Traditional walkers consist of a rigid frame with handles and legs which require the user to have enough upper limb strength to pick up the walker and move it forward one step at a time. Although this provides the most stable solution, the increase in energy expenditure and higher stresses seen at upper limb joints may be undesirable to certain users [3]. Rollators are often referred to as "rolling walkers" and includes the addition of wheels under its legs. This improves maneuverability, allows for normal gait patterns and eliminates the need to lift the walker. The use of wheels does impose certain drawbacks such as reduced stability, limited terrain use and the possibility of the walker rolling away from underneath [4]. Robotic walkers are used to address the many concerns found in both types of walkers.

Robotic walkers offer a safer experience for users by incorporating smart sensors and actuators to rollators. Implementation of such sensors allows for features such as steering control, velocity/acceleration limits and obstacle avoidance [5]. Areas of research in this field integrates techniques for mapping, localization and path planning [5].

This emerging technology does present its own challenges. As robotic walkers predict the intended action of the user, it is critical to design robust control systems to ensure user safety. Determining accurate timing and magnitude parameters of the system will allow for efficient traversal of the walker during turning or moving in a straight path. A key area in this challenge is designing around lateral motion stability control [6].

Accurate estimation of lateral velocity in control systems allow for improved safety of walkers by maintaining desired orientation and lateral trajectory. Current research in this area explores the estimation of lateral velocity of walkers using model-based approaches. This approach is limited given the difficulty of obtaining such model parameters through the use of instrumentation.

In the literature, several vehicle velocity estimation techniques have been suggested. Some methods estimate both longitudinal and lateral velocity simultaneously. For example, [7], [8], [9], [10] uses nonlinear observers to estimate both parameters. Furthermore, in [11] an algebraic approach based on numerical differentiation and diagnosis has been proposed. This does not rely on a tire model in order to obtain estimation which is seemingly more robust to system uncertainties and disturbances such as friction. It is important to highlight that in many of these approaches, the method was not validated through experimental testing or validation.

The literature also considers other approaches used to estimate velocities by using filter based methods such as the Extended Kalman Filter [12], [13], [14], [15] and the Unscented Kalman Filter [16], [17]. Experimental methods carried out in [18] use algorithms to evaluate the vehicle longitudinal velocity based on the measurements of wheel rotational speeds and vehicle acceleration. The main advantage of this approach is the minimal computation required, allowing for more accurate estimation results. Another empirical approach to estimate vehicle velocity was proposed by Jiang et al. by

using an adaptive nonlinear filter method with only encoders and without additional measurements of the vehicle [19].

Regarding walkers, Zhang et al. introduced a method to obtain lateral velocity using a model that requires physical parameters of the walker system. The study did not indicate experimental results and was only performed through simulations [6]. Wada et al. highlighted an experimental approach by implementing a caster mechanism with encoders to accurately measure the steering angle for velocity estimation [20]. However, the addition of this mechanism increases the mass of the system which is undesirable to users.

Farrelly et al. later proposed a kinematic based approach for estimation of lateral velocity in vehicles using inertial measurement units (IMU) and encoder data [21]. Implementing such methodology later revealed unexpected results that did not match those introduced in the paper. This is likely due to using expensive IMUs with greater accuracy and precision found in many vehicles.

Deep learning based approaches have become quite popular within the automotive industry due to their superior performance in complex environments. In [22] a deep neural network was employed to estimate vehicle orientations using a camera-based model. However, such a model is constrained by environmental conditions when compared to LIDAR based approaches. In [23], a method involving unsupervised approaches was implemented on trajectory clustering and modelling. Reinforcement learning based approaches have also gained traction in vehicle control, specifically for automated lane changing under unforeseen scenarios seen in [24].

A very popular approach is the Visual Inertial approach. There has been significant research done in this field with very promising results. Visual Inertial approaches are very common in mobile robots, drones and autonomous vehicles. A fundamental component of visual based methods is feature tracking [5]. While this is fairly easy for autonomous vehicles due to limited number of scenes it is expected to see (e.g, most roads have lane marks which can be used), and for drones due to its larger area coverage due to elevation (with high altitudes, you can see a lot more of the scene and have a better chance of tracking a feature), is is very difficult in Walker platforms. Furthermore, a lot of walkers are used in sensitive areas (retirement homes, hospitals), where vision based systems are not always welcomed due to privacy concerns. A research group in Stanford led by Fei-Fei has developed many algorithms for hospital vision systems who address the privacy concerns, however they come at great computational costs [25].

Recent advancements in machine learning have also enabled the use of sensor data to make accurate state estimates without the use of expensive equipment. Wang et al. proposed DeepSpeedometer, a model that uses a Long Short Term Memory (LSTM) based network to predict velocity using raw accelerometer and gyroscope data [26]. The proposed approach was able to outperform integration methods in all test cases with an average percent error of $< 0.5\%$ [26]. RoNIN is another project that utilizes machine learning approaches to estimate individual position and orientation using IMU data [27]. The approach investigates architectures based off LSTM, Residual Neural Network (ResNet) and Temporal Convolution Network (TCN) backbones.

The main contributions of this paper are as follows:

- Compare various methods for lateral velocity estimation
- Attempt to combine Learning method with Filter based method for improved estimation

II. Kinematics

The dynamics of the non-holonomic vehicle system is shown below with heading angle θ, track width *2R* and reference frame origin P_0.

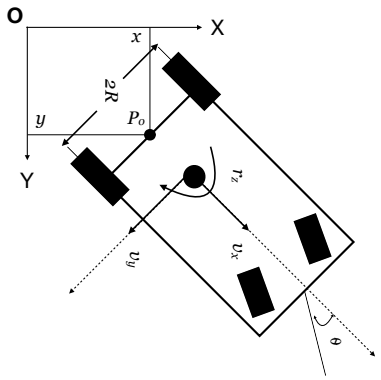

Fig. 1: Walker Dynamics

State observers of a dynamic system provide an estimate of the internal states which require measurements from the system inputs and outputs. The parameters of the model are the following; longitudinal and lateral accelerations a_x, a_y, yaw rate r_z and measurements of the vehicle longitudinal velocity, $v_{(meas)}$ to produce estimates of longitudinal and lateral velocities v_x, v_y. The kinematic model of the system is the following:

$$\dot{v}_x = v_y r + a_x \tag{1}$$

$$\dot{v}_y = -v_x r + a_y \tag{2}$$

In state space form, the kinematic model can be expressed as:

$$\dot{x} = \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \end{bmatrix} = A \begin{bmatrix} v_x \\ v_y \end{bmatrix} + B \begin{bmatrix} a_x \\ a_y \end{bmatrix} \tag{3}$$

$$A = \begin{bmatrix} 0 & r_z \\ -r_z & 0 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4}$$

A limitation of this model is that it can only function when yaw rate is non-zero as demonstrated in the observability matrix.

$$\mathcal{O} = \begin{bmatrix} 1 & 0 \\ 0 & r_z \end{bmatrix} \tag{5}$$

While the above method is very tempting to use, another huge issue is the IMU noises. IMU data can be very noisy, especially for cheap IMUs. IMU acceleration data is the most prone to error as well. Relationship between the IMU measurement and actual value can be modeled as:

$$w_t^m = w_t + b_t^w + \mathrm{w}_t^w \tag{6}$$

$$a_t^m = a_t + b_t^a + \mathrm{w}_t^a \tag{7}$$

where b_t^w, b_t^a are quasi-constant angular velocity and acceleration biases and w_t^w, w_t^a are zero-mean Gaussian noises. IMU biases are constantly changing due to various reasons such as temperature effect and can be difficult to estimate. Acceleration data noise is especially harmful as when integrated to find velocity, the errors gets amplified due to small time interval.

III. Integration Method

From relying solely on the kinematic model, the following state observer was developed by Farelly et al. [21]:

$$\begin{bmatrix}\dot{\hat{v}}_x \\ \dot{\hat{v}}_y\end{bmatrix} = (A - KC)\begin{bmatrix}\hat{v}_x \\ \hat{v}_y\end{bmatrix} + B\begin{bmatrix}a_x \\ a_y\end{bmatrix} + Kr_z \tag{8}$$

$$C = \begin{bmatrix}1 & 0\end{bmatrix}, K = \begin{bmatrix}2\alpha|r_z| \\ (\alpha^2 - 1)r_z\end{bmatrix} \tag{9}$$

where K is the estimator gain matrix and α is the observer design parameter. After tuning this value, it was found that α being close to 3 yielded the best results. In discretized form, assuming a zero-order hold, state estimates can be shown to be:

$$\begin{bmatrix}\hat{v}_{x,k+1} \\ \hat{v}_{y,k+1}\end{bmatrix} = [T(A-KC)+I]\begin{bmatrix}\hat{v}_{x,k} \\ \hat{v}_{y,k}\end{bmatrix} + T(B\begin{bmatrix}a_x \\ a_y\end{bmatrix} + Kr_z) \tag{10}$$

where T is the sampling period. This was kept at a constant $0.004s$ for all trials.

The main advantage of this method is that you are able to use it through the vehicle linear and non-linear handling region. Furthermore, unlike other model based approaches, this does not require any vehicle parameters which is the main reason behind exploring this method. Other approaches require vehicle parameters which are dependent on mass, which is constantly changing on a walker due to the human mass effect and the significance of this effect is substantial. It is possible to estimate these parameters but it comes at great computational complexity and requirement of additional mechanisms being added to the platform, thus increasing the cost of the Walker, and more importantly the mass which in undesirable for walkers due to its use by the elderly. Also for this, you will also need to characterize the robot tire cornering forces/coefficient, which is also changing depending on the surface it is being used in. Estimating this adds even more complexity to the model based approaches.

The beauty of the kinematic model is that it contains no physical parameters of the vehicle and hence is unaffected by changes in them. It also has low computational cost as well as simple tuning method. One drawback of the kinematic model is the requirement of reliable IMUs. Cheap IMU acceleration data can be very noisy and expensive IMUs are not really a viable option for cheap walkers. When integrated over and over again, the values tend to drift away from ground truth especially for higher accelerations and this error continues to build up. Lateral velocity estimation for this method can be improved by resetting the observer when the yaw raw or longitudinal velocity is under a certain threshold, but even with that it can still be unreliable.

Figures 2 and 3 show some preliminary results for this. As evident, the integration method works pretty well at low lateral velocities, but quickly drifts when velocity increases and held for longer amount of time. An important observation to be made is in Figure 3, the IMU tracks the lateral velocity very well for straight maneuvers, especially since the observer can be reset when the yaw rate is low.

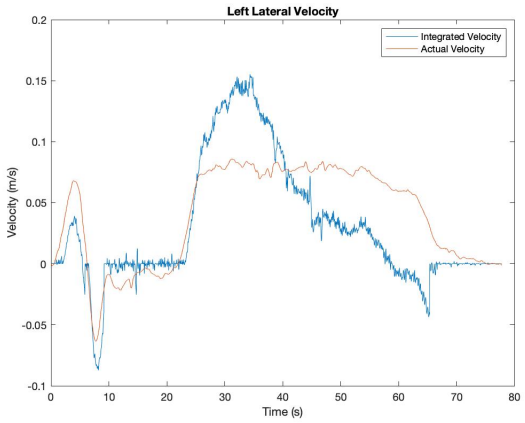

Fig. 2: Left Turn Lateral Velocity

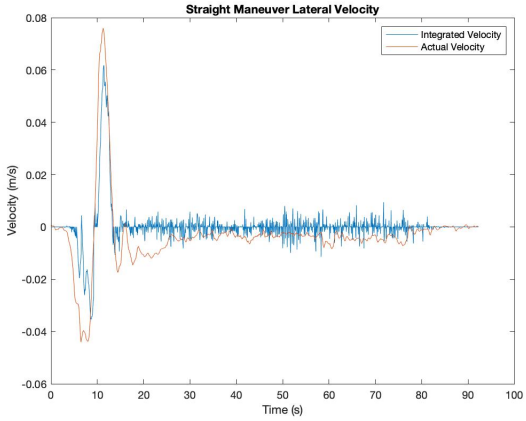

Fig. 3: Straight Lateral Velocity

It has been shown before in studies that accurate velocity estimation can be made with reliable IMUs. Due to this, we believe that if we can characterize the IMU better, we can potentially estimate this with fairly high accuracy. IMU characterization is a very difficult task to accomplish, hence we propose using learning based models to better estimate the lateral velocity.

IV. TCN

With the advancements in the the machine learning community, more and more projects have proposed learning networks for sensor data classification and regression problems.

Recurrent Neural Networks (RNNs) are the most common for time series problems as it uses outputs from the past to make predictions at the current time step t. The main issue with RNNs is its ability to carry information forward for long sequences. This is mainly due to the well known 'vanishing gradient problem'. Gradients are essentially values by which the weights of the network are updated. In the back propagation process during training, the gradients become too small where a change in the weight has essentially no impact on the output. LSTMs, a modification of the RNN has been proposed to tackle this issue. LSTMs are appealing as the hidden state also passes on relevant information from the data it has seen so far using an input gate and forget gate layer. However a major drawback of LSTMs is that they can take up a lot of memory to store the partial results of the various gates during training.

Recent work has shown that Convolutional Neural Networks (CNNs), also have the capability to achieve performance similar to that of RNNs [28]. It is now widely being used in audio synthesis and language level tasks. For this project, we explore Temporal Convolution Network (TCN).

TCNs have 2 distinguishing features, its casual convolutions and that the output can be of the same size as the input sequence [28]. Casual convolutions are convolutions that only uses data from time step t and before, which ensures that there is no information leakage from the future. TCN also uses a 1D fully convolutional layer with each hidden layer being the same length as the input layer and zero padding to ensure all layers forward are the same length as well. This way, we can achieve an output of same size as the input. TCN convolution can therefore be written as:

$$TCN = 1DFCN + CasualConvolutions \tag{11}$$

However, to represent TCNs with a long history size, a very deep network or large filters are required, both of which require large amounts of memory and is not a very feasible method. To tackle this, dilated convolutions have been suggested which allows a large receptive field. Dilated convolution can be given by Equation 12.

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i} \tag{12}$$

where d is the dilation factor and k is the filter size. A visual representation of how the dilation increases the receptive field can be seen in Figure 4. Therefore the receptive field can be increased by both increasing the dilation factor as well as having larger filter sizes.

The last component of TCNs are its residual connections. The residual block concatenates the input data with the transformations of the input and is given by:

$$o = Activation(x + \mathcal{F}(x)) \tag{13}$$

As the input undergoes large amounts of transformations especially with deeper networks the current time step data is often lost, but by concatenating we are able to learn the modifications to the original input. Before concatenating the input data is fed through a 1x1 convolution to match the size of the tensors as seen in Figure 5. In the residual block, each dilated casual convolutions are followed by a weighed normalization layer, a ReLU activation layer to add non linearity and a dropout layer to avoid over fitting of the data. After all the transformations is where the data is concatenated with the input and is seen in Figure 5.

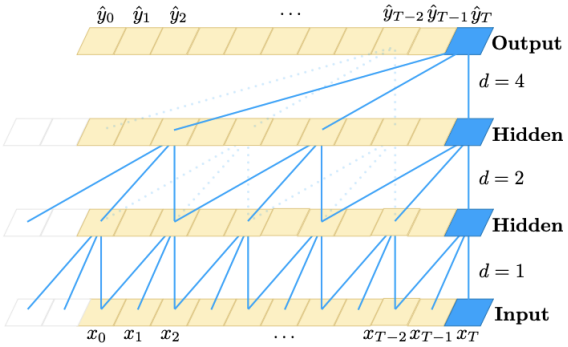

Fig. 4: Dilation Receptive Field [28]

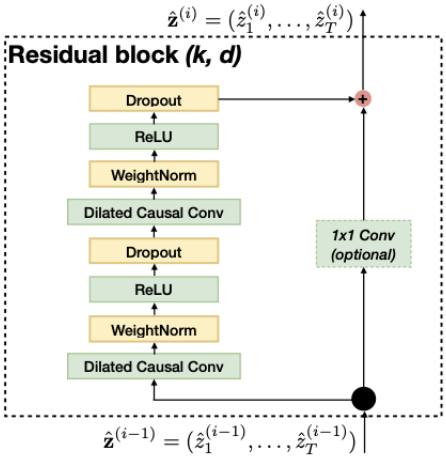

Fig. 5: Residual Block [28]

One of the main advantages of TCN over other RNN models is that the convolutions can be done in parallel due to the same filters being reused. Furthermore, the TCN models have a much better response to change in gradients and thus can take in long sequences of input unlike RNNs, which have shown to suffer from vanishing gradient problems. For training, less memory is also required compared to LSTMs since no partial results are stored from time step to time step [28].

TCNs also have some downsides. In TCNs the whole sequence has to be fed at each time step, and if the sequence gets very large, during testing memory can be used up very easily whereas in RNNs an input of fixed size encodes the past history and therefore is independent of the sequence length [28]. However, for our case, this is not a very big issue as typically for walker users, turns are over short periods of time. Once a turn in completed, the lateral velocity is expected to be 0, and hence the past input can be discarded and the observer can be reset.

TABLE I: Model Parameters

Parameter	Value
Dilation Factors	8, 16, 32, 64
Filters	64
Kernel Size	3
Dropout Probability	0.2
Optimizer	Adams
Loss Function	Mean Squared Error
Batch Size	500
Learning Rate	0.0005
Epochs	75
Validation Split	0.2

Lateral velocity estimation is crucial for walker controls, especially since it is very safety critical. Learning based models add significant computational cost over model based methods, however learning methods are expected to outperform model based approaches drastically. Furthermore, our current platform is already equipped with a computer that is able to supply the required power and hence, no additional cost is added to the development of the platform. Due to the nature of our system, the model also does not have to be very deep and thus, memory required is limited. Due to all these advantages, a learning model is proposed.

The network and training parameters used for our training is summarized in table I. Various models were trained and tested and the optimal model was selected.

V. Filters

State estimates frameworks for Kalman based filters are represented by a system equation and a measurement equation given by Equation 14.

$$x_{k+1} = F(x_k, u_k, v_k) \tag{14a}$$

$$y_k = H(x_k, n_k) \tag{14b}$$

where x_k is the system state, u_k is the control input, y_k is the observed measurement signal, v_k is the process noise with covariance Q_k and n_k is the measurement noise with the covariance R_k. An illustration of the block diagram is shown in Figure 6.

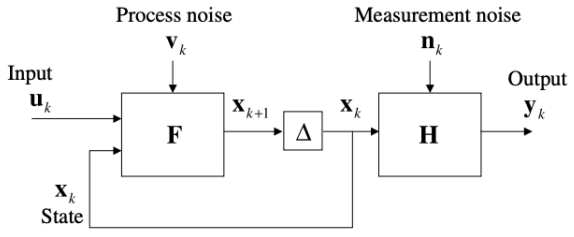

Fig. 6: Filter Diagram [29]

A. Kalman Filter

One of the most popular types of filter to remove noise from data is the Kalman Filter. In addition, it is widely used in fusing information coming from multi-sensor configurations. The utility comes from being able to provide optimal state estimates, even if only noisy measurements are available or the system model is inaccurate. The algorithm is recursive and is comprised of two steps: the first being where the state $\hat{x}_{k|k-1}$ is predicted using the state transition matrix from Equation 14a. The error covariance matrix P, which provides a measure of confidence in the estimate, is updated using:

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q \tag{15}$$

At this point, a measurement is introduced to improve the prediction. For longitudinal velocity, the average rotational velocities from the wheel encoders was used as shown below:

$$v_{x,meas} = \frac{w_l + w_r}{2} R_w \tag{16}$$

As for lateral velocity, predictions generated from the TCN outlined earlier will serve as the measurements. When only using the v_x measurements, the observation matrix H will be identical to C in Equation 9. This will change to a 2 x 2 identity matrix when v_y is also used to maintain dimensional agreement with the state vector.

With the latest measurement y_k, the state estimate and error covariance are updated once again. Also, the Kalman gain K_k is calculated, which helps determine how much weight is given to the measurement to change $\hat{x}$ and P.

$$K_k = P_{k|k-1} H^T (H P_{k|k-1} H^T + R)^{-1} \tag{17}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (y_k - H \hat{x}_{k|k-1}) \tag{18}$$

$$P_{k|k} = H^T (I - K_k H) P_{k|k-1} \tag{19}$$

A limitation of the Kalman Filter is that it is meant for dealing with linear systems. Any slightly non-linear models can be approximated with linearization in the Extended Kalman Filter. Another limitation comes from the assumption that the variances are Gaussian in nature, which is not always the case. This is relevant since these assumptions may not hold for the lateral velocity measurements from the TCN, which come from a non-linear model with a non-Gaussian error distribution. Thus, the filter may not yield the best results.

B. Unscented Kalman Filter

For our case, we use a learning based model to predict the lateral velocity of the system. This method is very susceptible to the sensor noise and the output we see is very noisy as seen in the results section. To improve our estimate, we use the model based output as a sensor measurement and then use our knowledge of the system dynamics to filter out noise. The original Kalman Filter can be used here, however it is not optimal for non-Gaussian probability Density Functions (PDF). As our prediction is made using a learning network which adds non-linearity to the measurement and the noise can also not assumed to be Gaussian, we propose using the Unscented Kalman Filter (UKF). Particle filters do a much better job at representing the non-Gaussian noise, however come at the expense of computational complexity. Particle filters are based on Monte Carlo and uses all the particles from the past to represent the probability densities. The UKF

is a particle like which uses a carefully chosen set of sample points (sigma points) of size $2n+1$ where n is the number of states. These sample points capture the true posterior mean and covariance accurately to at least the second-order for non Gaussian data [29] and third order for Gaussian data. UKF has been particularly preferred choice for machine learning approaches. It has been widely used for state estimation, parameter estimation and dual-estimation. For the purpose of this project, it is only used for state estimation.

Considering a state x of size n where x has a mean of $\bar{x}$ and a covariance of P_x, the statistics of y can be calculated using a matrix of size $2n+1$ sigma vectors $\mathcal{X}$ using the following:

$$\mathcal{X}_{0,k-1} = \bar{x}_{k-1} \tag{20}$$

$$\mathcal{X}_{i,k-1} = \bar{x}_{k-1} + (\sqrt{(L+\lambda)P_{k-1}})_i, i = 1...L$$
$$\mathcal{X}_{i,k-1} = \bar{x}_{k-1} - (\sqrt{(L+\lambda)P_{k-1}})_{i-L}, i = L+1...2L$$

where λ is a scaling parameter calculated using

$$\lambda = \alpha^2(L+\kappa) \tag{21}$$

where α is a constant which determines the spread of the sigma points around $\bar{x}$. This value is typically set to a small positive constant. κ is a another scaling parameter and for our case it is set to 0 which is very common to do. β is used to use the prior knowledge of the distribution. For the prediction step, the points are propagated through the state function and then a priori state and covariance matrix estimate is made using equations,

$$\hat{x}_k^- = \sum_{n=0}^{2L} W_i^{(m)} \mathcal{X}_{i,k|k-1} \tag{22}$$

$$P_k^- = \sum_{n=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1} - \hat{x}_k^-][\mathcal{X}_{i,k|k-1} - \hat{x}_k^-]^T + Q_k \tag{23}$$

where the weights are calculated using:

$$W_0^{(m)} = \frac{\lambda}{L+\lambda} \tag{24}$$

$$W_0^{(c)} = \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta) \tag{25}$$

$$W_i^{(c)} = W_i^{(m)} = \frac{1}{2(L+\lambda)} \tag{26}$$

We then propagate $\mathcal{X}$ through the measurement model and then calculate the the mean and covariance using:

$$y_k^- \approx \sum_{n=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1} \tag{27}$$

$$P_{yy,k} \approx \sum_{n=0}^{2L} W_i^{(c)} [\mathcal{Y}_{i,k|k-1} - \bar{y}_k][\mathcal{Y}_{i,k|k-1} - \bar{y}_k]^T + R \tag{28}$$

$$P_{xy,k} \approx \sum_{n=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1} - \bar{x}_k^-][\mathcal{Y}_{i,k|k-1} - \bar{y}_k]^T \tag{29}$$

Finally, we update our estimate using the equations:

$$\mathcal{K}_k = P_{xy,k} P_{yy,k}^{-1} \tag{30}$$

$$x_k = \hat{x}_k^- + \mathcal{K}_k(y_k - y_k^-) \tag{31}$$

$$P_k = P_k^- - \mathcal{K}_k P_{yy,k} \mathcal{K}_k^T \tag{32}$$

where κ is the gain factor.

As seen above, the main advantages of using the UKF method is that it is able to represent the statistics using only a small set of sample points and is a better approximation of it. An illustration of the approximations are shown in Figure 7 by Wan et. al for the actual (a), first order linearization (b) and UT methods (c) respectively.

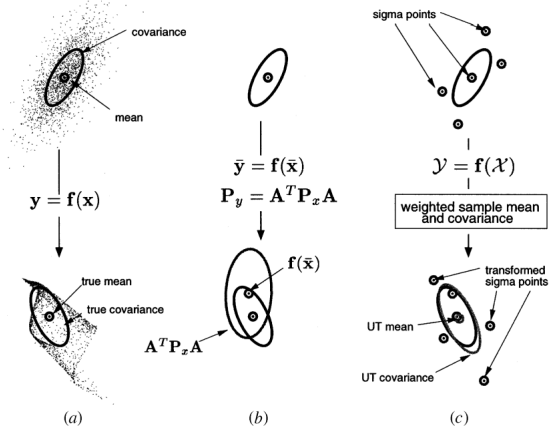

Fig. 7: UT Transformation [29]

VI. Results

A. Data Collection

An existing walker was modified with a 9 DOF IMU and a brushless DC Hub motor with built-in encoder and hall effect sensors as seen in Figure 8. The IMU is mounted symmetric to the y-axis. Both the IMUs and the encoders are connected to an Arduino which collects the data through serial communication with a laptop at 250Hz.

Fig. 8: Walker Platform

For data collection, trials were performed at the University of Waterloo's RoboHub. The RoboHub is equipped with 12

TABLE II: Root Mean Square Errors

Method	v_x RMSE	v_y RMSE
Integration	0.2834	0.2416
TCN	-	0.0295
KF	0.0196	0.2342
TCN + KF	0.0274	0.0257
TCN + UKF	0.0188	0.0110

Vicon Cameras, which were used to collect ground position data. The Vicon cameras are seen as state of art for position tracking and can track with very high accuracy. Three different types of trials were conducted where users were asked to either 1) walk straight with the walker, 2) walk straight and take a left turn or 3) Walk straight and take a right turn. Users were free to walk in their normal gait patterns, however were encouraged to walk at different speeds and turn with different turning radii for the various trials.

The Arduino data and the Vicon data were merged offline. The Vicon data was first passed through a low pass filter with a τ of 0.05. The data was then differentiated to find velocity and transformed using Equation 33 to find the velocities in the local coordinate system.

$$R(\theta) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{33}$$

The 2 sets of data were then merged manually. The Arudino data position has a spike when you initially start moving, which is matched to the same spike in the Vicon data. Although this is not the most accurate method of merging the data, it is a fair assumption to make that the merged data is within 0.1 seconds of each other, and as the walker is a fairly slow system, this small time is not expected to have a large difference.

B. Evaluation

For evaluation of the different methods, we use the Root Mean Square Error (RMSE) which is given in Equation 34.

$$RMSE = \sqrt{\frac{\sum_{k=0}^{N}(\hat{x}_k - x_k)^2}{N}} \tag{34}$$

where $\hat{x}_k$ is the state estimate, x_k is the actual state and N is the total number of time steps.

C. Performance

Table II summarizes the results for the various methods that were attempted for this project. As seen from the table, our proposed method of using a learning based model augmented with the Unscented Kalman filter outperforms all the other methods significantly.

To further investigate the advantages and disadvantages of the various methods, we will be using 2 of the test trials, one left turn trial and one right turn trial. The data for these trials can be seen in Figures 9 and 10 respectively. Note that the data shown is passed through a low pass filter with $\tau = 0.05$ but in reality, the raw data is fed into the network and the filters. The raw data itself is very noisy and it is very difficult to see the trends otherwise.

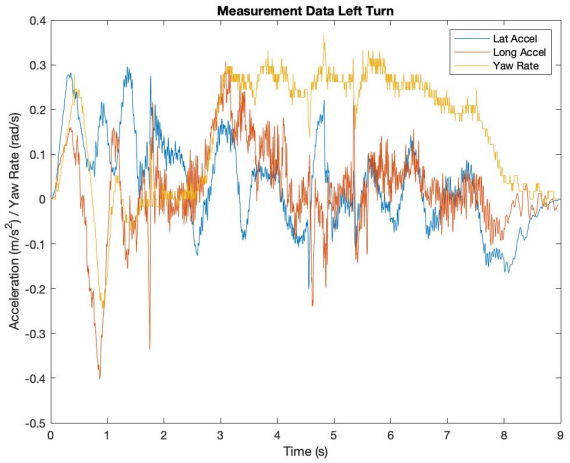

Fig. 9: Left Turn IMU Measurements

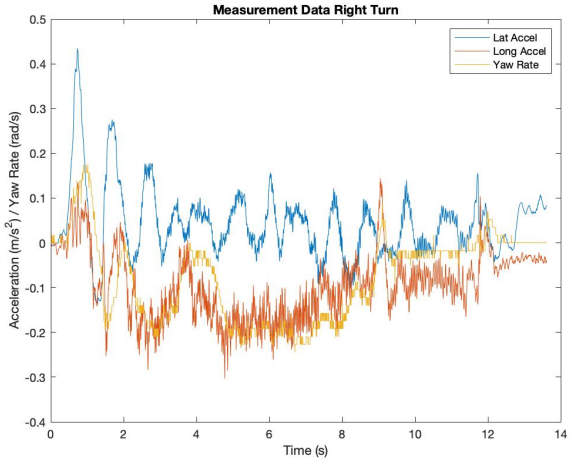

Fig. 10: Right Turn IMU Measurements

1) Integration Method: Figures 11 - 14 shows results for both lateral and longitudinal velocity for a left turn trial and a right turn trial. As seen from the graphs, the estimates are not very accurate and deviate from the true values very quickly. The gain factors were tuned for optimal performance. This is as expected as the integration method relies heavily on the IMU data. For our case, the encoder data and IMU r_z are fairly accurate whereas the acceleration data is very noisy. At each time interval, a new estimate is made using the past estimate and the current measurements and hence as time goes along, the estimate gets worse due to the noisy data. This method can be improved by having an expensive IMU with much better noise properties. Another method to prevent drift is resetting the observer when the longitudinal velocity or yaw rate is under a certain threshold.

One important observation to make here is that the longitudinal velocity estimates are also very inaccurate. While there is no way to measure lateral velocity (completely estimated), there is a way to measure the longitudinal velocity fairly accurately. Assuming no slip, the longitudinal velocity can be estimated using Equation 16. As we have a fairly good

estimate of the longitudinal velocity, the integration method is sub-optimal as the noise is additive. In situations like this, Kalman filters are expected to do a much better job due to its ability to combine the Gaussian error distributions from the state transitions and measurements and generate an estimate with a narrower distribution.

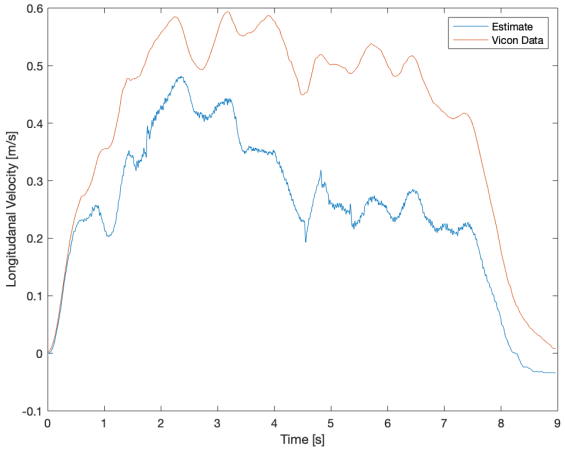

Fig. 11: Integration Method - Left Turn v_x Estimate

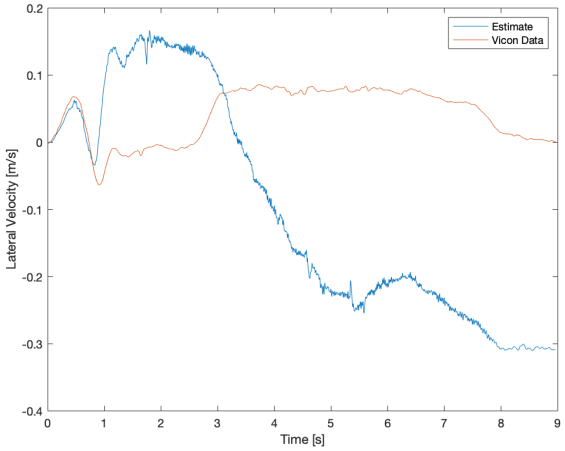

Fig. 12: Integration Method - Left Turn v_y Estimate

2) Kalman Filter: Figures 15 - 18 shows the results for the same trials as above but using a Kalman filter for its estimates. As expected, there is a drastic improvement in the v_x estimate. Not only is the noise not additive, but the Kalman filter also filters the noise in the measurements and thus improving the performance. However, as seen from the graphs, v_y estimate continues to be very inaccurate due to the noisy measurements. As a matter of fact, it is worse than the integration method as the gain factor is not manually tuned and since there is no measurement for v_y, the estimates continue to blow up. The lateral velocity estimate can be improved a bit by resetting the observer when yaw rate or velocity is under a certain threshold, however, even with that the performance is not expected to be up to the required standard. Due to this, learning based models

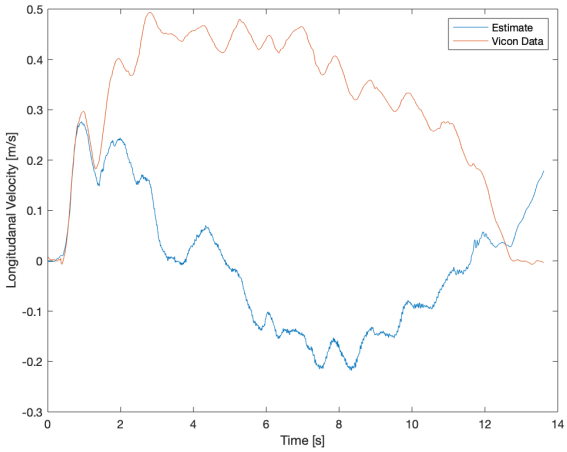

Fig. 13: Integration Method - Right Turn v_x Estimate

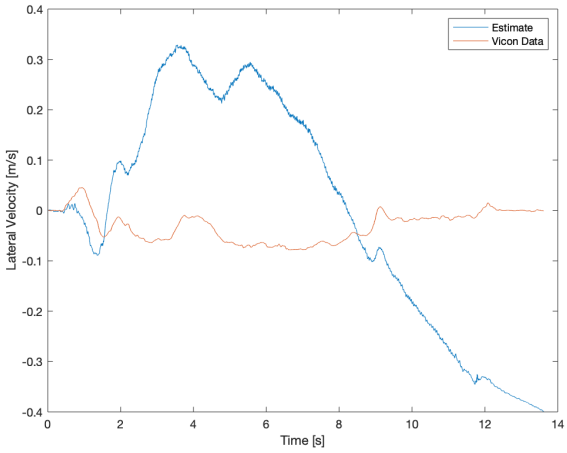

Fig. 14: Integration Method - Right Turn v_y Estimate

are proposed as it is predicted that the learning based model can do a better job at modeling the IMU noises and can also use the vehicle kinematic model to make its predictions.

3) TCN: TCN was used only for the lateral velocity estimation v_y since v_x can be estimated to acceptable accuracy using Kalman Filter as seen from above. Figures 19-20 show the v_y results for the same trials as above. As seen, TCN (RMSE= 0.0288) is a drastic improvement over the integration method (RMSE= 0.2379) and KF method (RMSE= 0.3193). This is as expected as it is very difficult to model cheap IMUs. Factors such as temperature and location of mounting can also lead to additional drift on top of the already noisy data.

One important thing to observe here is that the prediction trends seem to follow that of the actual state, however the prediction is very noisy. These graphs look similar to sensor measurements such as IR sensors with zero-mean Gaussian noise. Just by looking at the graphs, it is assumed that some sort of filtering to this can improve the estimate. We can use the the prediction as a 'sensor measurement' for v_y and feed that into the Kalman filter to filter noises.

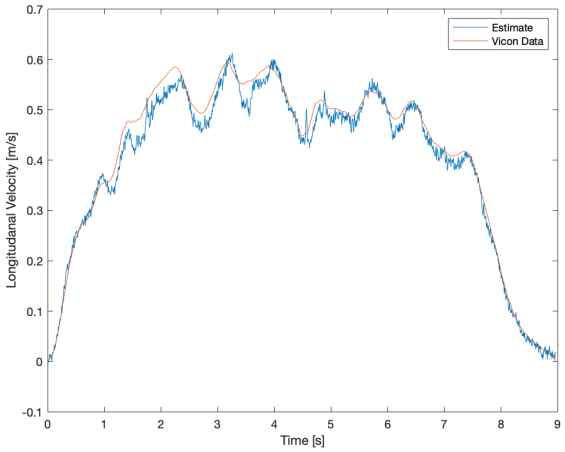

Fig. 15: Kalman Filter - Left Turn v_x Estimate

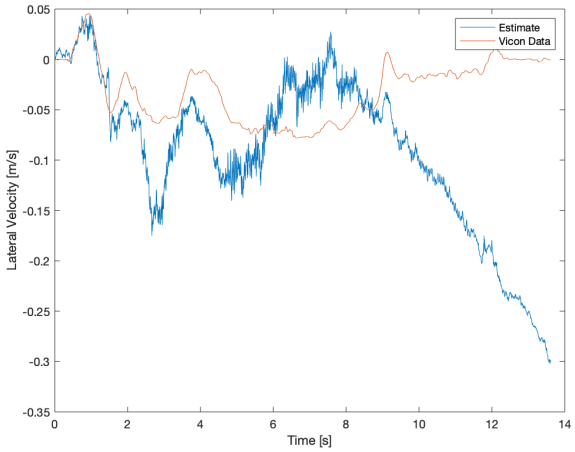

Fig. 18: Kalman Filter - Right Turn v_y Estimate

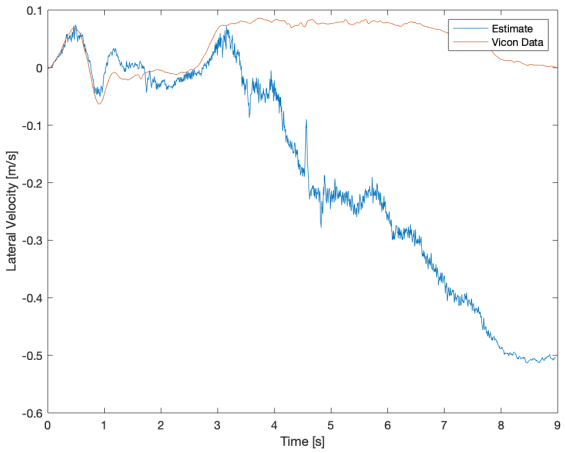

Fig. 16: Kalman Filter - Left Turn v_y Estimate

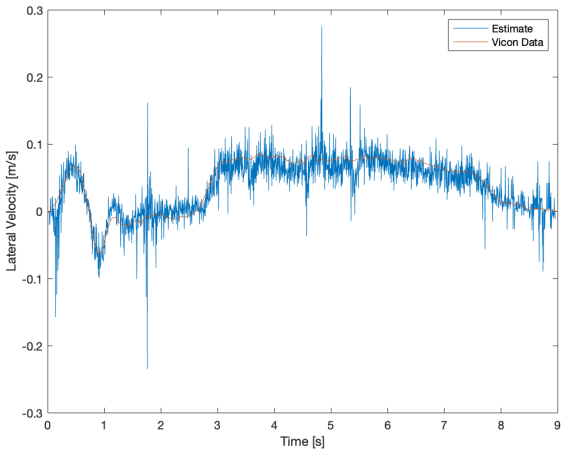

Fig. 19: TCN - Left Turn v_y Estimate

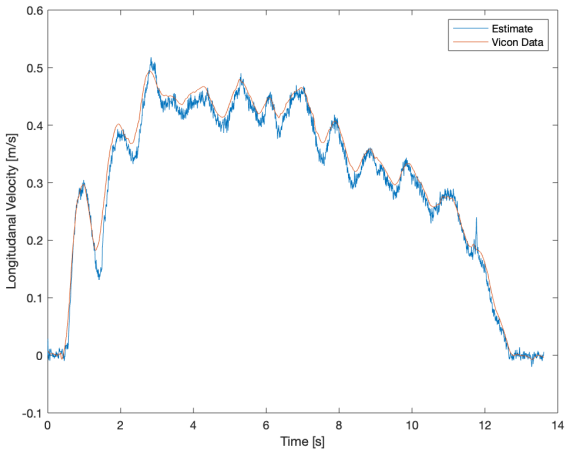

Fig. 17: Kalman Filter - Right Turn v_x Estimate

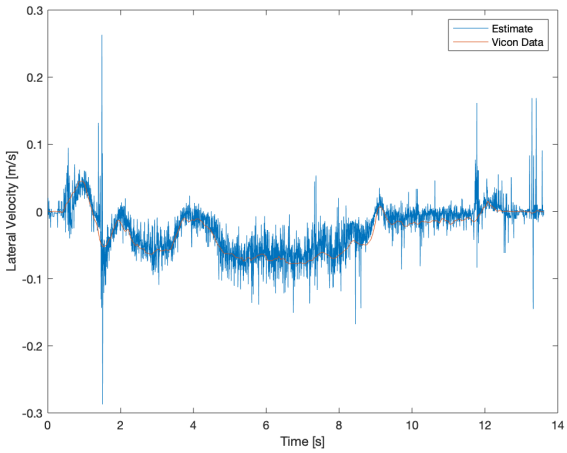

Fig. 20: TCN - Right Turn v_x Estimate

4) TCN w/ KF: Figures 21 - 22 show the results for the augmented method. One of the main benefits of this method is it is able to reject outliers using the system model knowledge. This augmented method shows an improvement over the pure TCN method (RMSE = 0.0257 vs 0.0295). Looking closely at the graphs, we can see that the filtered method has a smoother trajectory as we would expect from real systems. One flaw in this method is that we are assuming Gaussian noise for our 'sensor' measurement as well as assuming that it is a linear function. However in reality, our measurement function y is a non-linear function of the IMU measurements and the state x_k. Furthermore, zero mean Gaussian noise cannot be assumed due to the high non-linearity of the learning network. By challenging those assumptions, we propose using particle like filters since they do a much better job at estimating the true measurement statistics. UKF in particular is able to approximate at least to the 2^{nd} order accuracy for non-Gaussian noise.

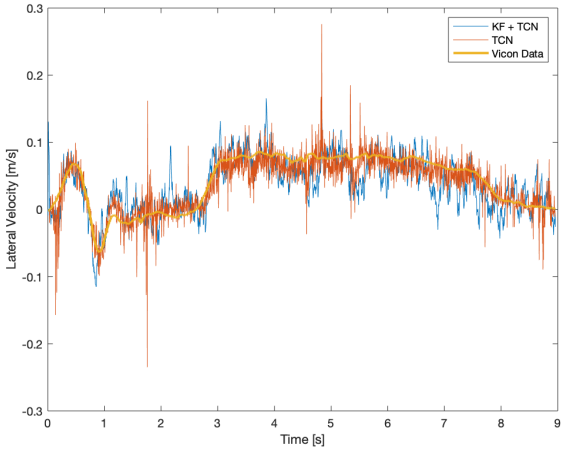

Fig. 21: KF + TCN - Left Turn v_y Estimate

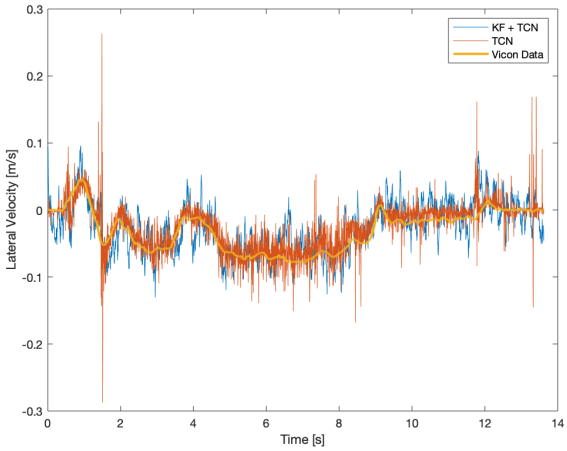

Fig. 22: KF + TCN - Right Turn v_y Estimate

5) TCN w/ UKF: Figures 23 - 24 show the results for augmenting the learning based model with UKF. As seen from the figures, this method shows substantial improvement (RMSE = 0.0110) over the TCN method (RMSE = 0.0295) and augmenting using KF (RMSE = 0.0257). This tells us that our belief of non-Gaussian noise was in fact correct. It is also evident that UKF is able to approximate the true covariance and mean much better than the KF method.

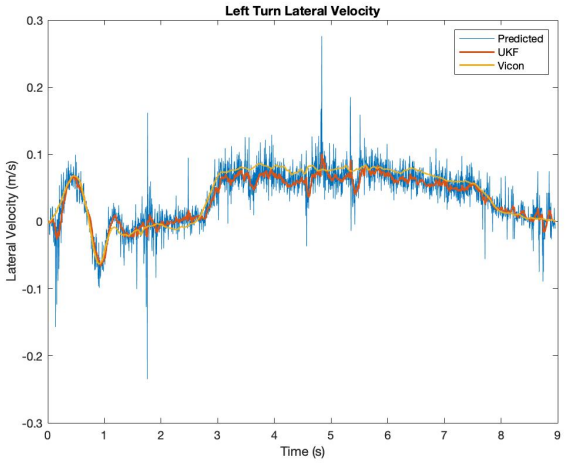

Fig. 23: TCN +UKF - Left Turn v_x Estimate

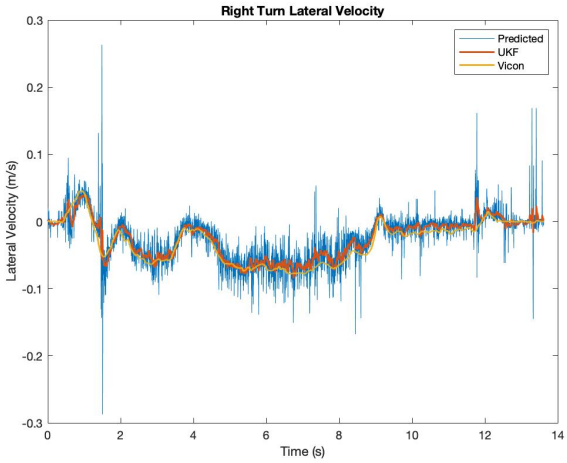

Fig. 24: TCN +UKF - Right Turn v_x Estimate

VII. TCN Ablation Study

Table III summarizes how different features affect the network. For this study, a smaller dataset was trained with same parameters as above but for only 30 epochs. 3 different networks with the following features were trained:

- Raw Features (RF): a_x, a_y, r_z, w_l, w_r for a given time window were directly inputed into the network without any filtering beforehand.
- Raw Features with temporal finite differences (RF+ dT): As temporal relationship is important in system statistics, the temporal difference in the raw measurement signals were taken and also fed into the network

TABLE III: Features

	RMSE
Raw Features	0.1237
RF + dT	0.1728
RF + v_x + diff	0.1198

- Raw Features with difference in wheel speeds and long. velocity (RF + v_x + diff): When turning, there is a difference between the wheel speeds ($diff = w_l - w_r$) of the platform which is an important indicator. Furthermore both the v_x and diff features have a direct relationship to the robot v_y.

There are 2 important observations to make here. First of all, meaningless features can affect the network negatively. Although temporal relationships are important, TCN is expected to already extract these temporal relationships on its own. By explicitly adding it into the training set, we actually see a negative impact. Secondly, although TCNs are expected to extract features on its own, there are some features it can not extract on its own and by having expert knowledge about the system dynamics and explicitly adding them in, we can improve the performance as seen from the table. Through this small study, it is evident that feature selection is an integral part of any learning based algorithms.

VIII. Conclusion

A more realistic stabilization control system for robotic rollators was explored by attempting to achieve more accurate predictions for lateral velocity. Existing techniques such as the Integration Method, Kalman Filter, Unscented Kalman Filter and TCN were compared to see which performed best. In addition, learning based and filter based methods were fused to check if any visible improvements would be made. It was found that the the Integration Method had the worst results as expected. The Kalman Filter by itself did not performed significantly better for v_x with the introduction of encoder measurements. This same improvement was not seen in v_y until the filter was combined with the predictions from the TCN. Although the RMSE was relatively the same, the Kalman Filter removed the outliers that were present when using the TCN by itself. The UKF-TCN combination gave the lowest error with it's ability to track non-linear systems. In the future, it would be worth investigating other techniques, especially ones that can handle non-linearity such as the particle filter. At the same time, it would be necessary it also study computational costs associated with each method in order to assess if the performance is justifiable.

References

[1] J.-P. Michel, *WHO world report on Ageing 2015*, 11 2015.

[2] R. Alsnih and D. Hensher, "The mobility and accessibility expectations of seniors in an aging population," *Transportation Research Part A: Policy and Practice*, vol. 37, pp. 903–916, 02 2003.

[3] S. Mohammed, J. Moreno, K. Kong, and Y. Amirat, *Intelligent Assistive Robots*. Springer Internnational PU, 2015.

[4] J. Stevens, K. Thomas, L. Teh, and A. Greenspan, "Unintentional fall injuries associated with walkers and canes in older adults treated in u.s. emergency departments," *Journal of American Geriatrics Society*, vol. 57, pp. 1464–1469, 2009.

[5] A. Morris, R. Donamukkala, A. Kapuria, A. Steinfeld, J. Matthews, J. Dunbar-Jacob, and S. Thrun, "A robotic walker that provides guidance," in *2003 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 2003, pp. 25–30.

[6] X. Zhang, J. Li, Z. Hu, W. qi, L. Zhang, Y. Hu, H. Su, G. Ferrigno, and E. De Momi, "Novel design and lateral stability tracking control of a four-wheeled rollator," *Applied Sciences*, vol. 9, p. 2327, 06 2019.

[7] L. Imsland, H. Grip, T. Johansen, T. I. Fossen, J. Kalkkuhl, and A. Suissa, "Nonlinear observer for vehicle velocity with friction and road bank angle adaptation-validation and comparison with an extended kalman filter," in *SAE Technical Paper*, 2007.

[8] H. F. Grip, L. Imsland, T. A. Johansen, J. C. Kalkkuhl, and A. Suissa, "Vehicle sideslip estimation design, implementation, and experimental validation," in *IEEE Control Systems Magazine*, vol. 29, Oct 2009, pp. 36–52.

[9] L.-H. Zhao, Z.-Y. Liu, and H. Chen, "Design of a nonlinear observer for vehicle velocity estimation and experiments," in *IEEE Transactions on Control Systems Technology*, vol. 19, May 2011, pp. 664–672.

[10] M. Oudghiri, M. Chadli, and A. El Hajjaji, "Lateral vehicle velocity estimation using fuzzy sliding mode observer," in *Proceedings 15th Mediterranean Conference on Control and Automation*, 2007, pp. 1–6.

[11] J. Villagra, B. d'Andrea-Novel, M. Fliess, and H. Mounier, "Estimation of longitudinal and lateral vehicle velocities: An algebraic approach," in *Proceedings IEEE American Control Conference*, 2008, pp. 3941–3946.

[12] H. Guo, Z. Wu, B. Xin, and H. Chen, "Vehicle velocities estimation based on mixed ekf," in *Proceedings Chinese Control and Decision Conference*, 2011, pp. 2030–2035.

[13] H. Guo, H. Chen, F. Xu, F. Wang, and G. Lu, "Implementation of ekf for vehicle velocities estimation on fpga," in *IEEE Transactions on Industrial Electronics*, vol. 60, Sep. 2013, pp. 3825–3835.

[14] L. R. Ray, "Nonlinear state and tire force estimation for advanced vehicle control," in *IEEE transactions of Control System Technology*, vol. 3, Mar. 1999, pp. 1–8.

[15] L. Chu, Y. Shi, Y. Zhang, H. Liu, and M. Xu, "Vehicle lateral and longitudinal velocity estimation based on adaptive kalman filter," in *Proceedings IEEE 3rd ICACTE*, 2010, pp. 325–329.

[16] X.-Y. Zong and W.-W. Deng, "Study on velocity estimation for four-wheel independent drive electric vehicle by ukf," in *Proceedings 5th International Conference on Control, Mechatronics and Automation*, 2013, pp. 1111–1114.

[17] L. Chu, Y. Shi, and M. Liu, "Vehicle lateral and longitudinal velocity estimation based on unscented kalman filter," in *Proceedings IEEE 2nd International Conference on Educational Technology and Computers*, vol. 3, 2010, pp. 427–432.

[18] M. Tanelli, S. Savaresi, and C. Cantoni, "Longitudinal vehicle speed estimation for traction and braking control systems," in *Proceedings IEEE International Conference on Control Applications*, 2006, pp. 2790–2795.

[19] F. Jiang and Z. Gao, "An adaptive nonlinear filter approach to the vehicle velocity estimation for abs," in *Proceedings IEEE International Conference on Control Applications*, 2000, pp. 490–495.

[20] M. Wada, K. Ichiryu, T. Iguchi, and R. Yoshida, "Design and control of an active-caster electric walker with a walk sensing system (smart walker)," in *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, July 2016, pp. 258–263.

[21] J. Farrelly and P. Wellstead, "Estimation of vehicle lateral velocity," in *Proceeding of the 1996 IEEE International Conference on Control Applications IEEE International Conference on Control Applications held together with IEEE International Symposium on Intelligent Control*, Sep. 1996, pp. 552–557.

[22] J. Park, Y. Yoon, and J. Park, "Deep learning-based vehicle orientation estimation with analysis of training models on virtual-worlds," in *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 2019, pp. 1–7.

[23] S. Mozaffari, O. Y. Al-Jarrah, M. Dianatti, P. Jennings, and A. Mouzakitis, "Deep learning-based vehicle behaviour prediction for autonomous driving applications: A review," 2019, pp. 1–7.

[24] P. Wang, C.-Y. Chan, and A. de la Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *IEEE Intelligent Vehicles Symposium*, June 2018, pp. 1–7.

[25] S. Yeung, F. Rinaldo, J. Jopling, B. Liu, R. Mehra, N. L. Downing, M. Guo, G. M. Biaconi, A. Alahi, J. Lee, B. Campbell, K. Deru, W. Beninati, L. Fei-Fei, and A. Milstein, "A computer vision system for deep learning-based detection of patient mobilization activities in the icu," in *npj Digital Medicine*, vol. 2, 2019, p. 11.

[26] Q. Wang, Y. Gu, J. Liu, and S. Kamijo, "Deepspeedometer: Vehicle speed estimation from accelerometer and gyroscope using lstm model," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017, pp. 1–6.
[27] H. Yan, S. Herath, and Y. Furukawa, "Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, and new methods," 2019.
[28] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018.
[29] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 2000, pp. 153–158.