Simulation-Based Engineering Lab
University of Wisconsin-Madison
Technical Report TR–2022–02

Using random walks to simulate GPS sensing for applications in robotics and autonomous vehicles

Stefan Caldararu, Harry Zhang, Ishaan Mahajan, Thomas Hansen, Shouvik Chatterjee, Nevindu Batagoda, Radu Serban, Dan Negrut

Department of Mechanical Engineering, University of Wisconsin – Madison

March 27, 2023

Abstract

We propose a new GPS model that produces synthetic data that mimics more accurately real GPS data. An analysis of current GPS models is presented, and a new model based on Concentration Gradient Random Walks in the second derivative of the measurement is proposed. This model is implemented in the Chrono::Sensor [1] module, and utilized in the ART/ATK [2] project to demonstrate its capabilities. In the future this model will be used in applications for mitigating the sim-to-real gap. Specifically, it will be used in a Kalman Filter application to measure performance between simulated and real state estimation, as well as an MPC formulation to drive through waypoints.

Keywords: robot simulation, autonomous vehicle simulation, sensor simulation, GPS simulation, random walk

Contents

1 Introduction

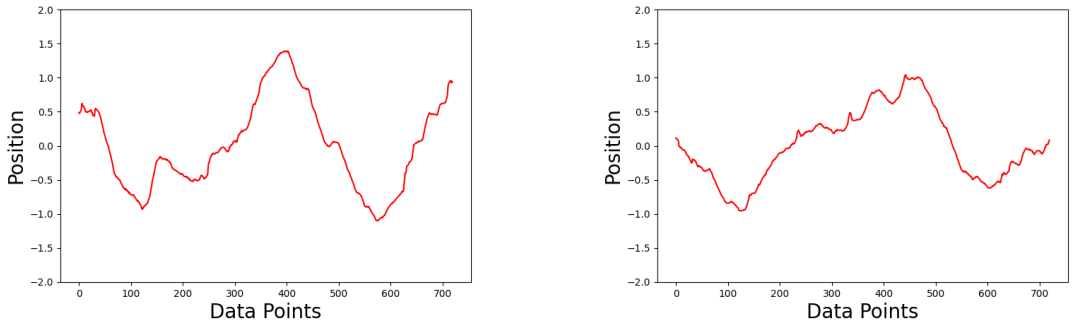
In this technical report we present a new GPS model in Chrono::Sensor based on Random Walks. The report is structured as follows. Section 2 shows GPS data and gives some background on modeling GPS data. Section 3 gives a GPS model based on Concentration Gradient Random Walks. Section 4 presents the specific software implementation in Chrono [3, 4] and parameters to be passed into the GPS model. Section 5 presents numerical experiments for the new GPS model. Section 6 provides a summary and future steps.

2 Background

GPS sensors are commonly used as a baseline localization sensor for autonomous vehicles. When modeling GPS sensors to be used in robot simulation, the majority of models assume a Gaussian Normal distribution. However, in reality GPS data does not look like this. An analysis of GPS data is presented in this section.

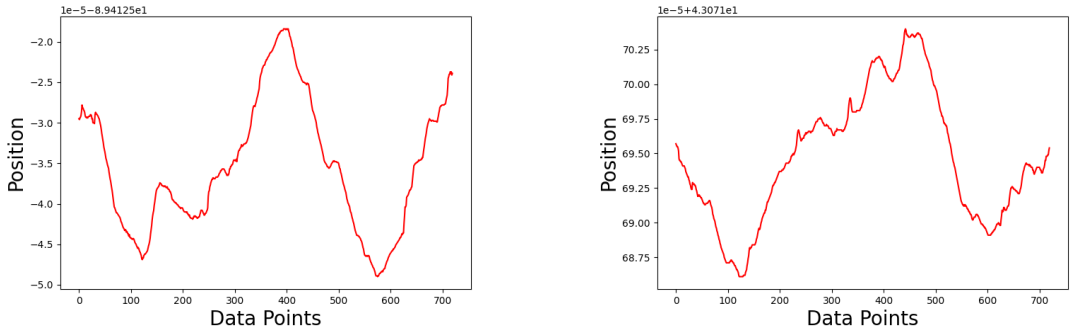
The following GPS data was collected on Parking Lot 17 in the UW Engineering campus. Data was collected with a stationary sensor at nine different locations. The nine different points were obtained by creating an orthogonal 3 by 3 grid, with every point being separated from its grid neighbor by a distance of 1.5 meters.

Data was collected at each location for approximately five minutes. Data was registered at around 3.3 Hz obtaining about 1000 data points over each five-minute span.



(a) X Coordinates (b) Y Coordinates

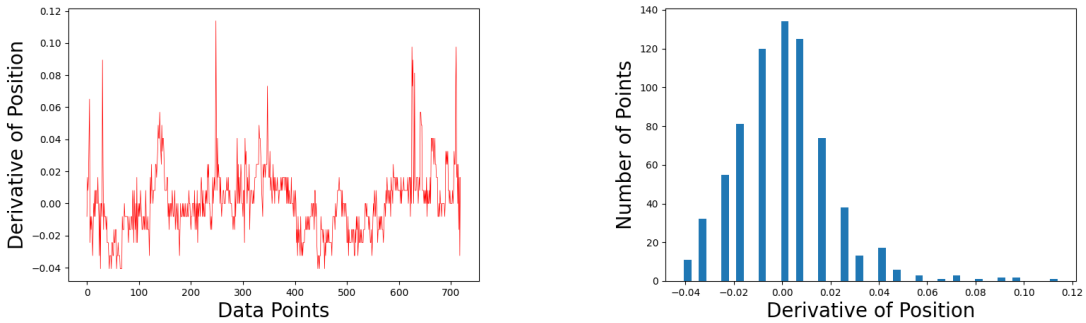
Figure 1: GPS data in Meters for a single point



(a) Longitude Coordinates (b) Latitude Coordinates

Figure 2: GPS data in Latitude/Longitude for a single point

Figure 1 shows data converted into meters in Cartesian coordinates at a given point, and Figure 2 shows the same data in the Geographic coordinate system. Analyzing the data, we observed that the approximate standard deviation across all points was 0.5 meters - although the distribution does not look normal. There is a clear dependence on previous points. When we take the first derivative of this measurement, we observe a normal distribution. Figure 3a shows the first derivative of Figure 1a, and Figure 3b shows the distribution of the values observed.



(a) Derivative of data collected (b) Histogram

Figure 3: Derivative of GPS data

3 Model Description

3.1 Model Derivation

The proposed GPS model uses a concentration gradient random walk in the second derivative of our measurement distribution. There is a separate Random Walk in each dimension. There are two key parts to this model. We want to have the normal distribution in the second derivative to increase smoothness, and the concentration gradient to maintain a zero mean in our measurements.

$$p_{t+1} = p_t + v_t v_{t+1} = N(m, \sigma) \tag{1}$$

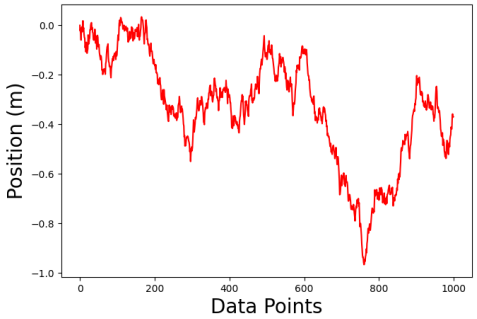


Figure 4: Random Walk with Normal Distribution in first derivative

A regular random walk follows a distribution as described in Equation 1. It maintains a dependence on the previous measurement, and has a normal distribution in the first derivative. This produces very choppy movements, as shown in Fig. 4. To smooth out this data, we move the normal distribution down to the second derivative.

$$\mathbb{E}(p_{t+1}) = p_t \tag{2}$$

In addition to this, another issue we can observe from Fig. 4 is that the mean of our distribution will not be 0. We can observe this in accordance with Equation 2. We want the expected value of the rest of our distribution starting at any point to approach 0, i.e. the true measurement. This is why we want to introduce a concentration gradient. As our measurements drift from the mean, we want to make it more likely that they will return towards a noise value of 0. Following these two problems, we propose the model described in Eq. 3.

$$\begin{aligned} m_{t+1} &= \frac{-p_t}{p_{max}} \\ a_{t+1} &= N(m_{t+1}, \sigma) \\ v_{t+1} &= v_t + a_{t+1} \\ p_{t+1} &= p_t + v_t + a_{t+1} \end{aligned} \tag{3}$$

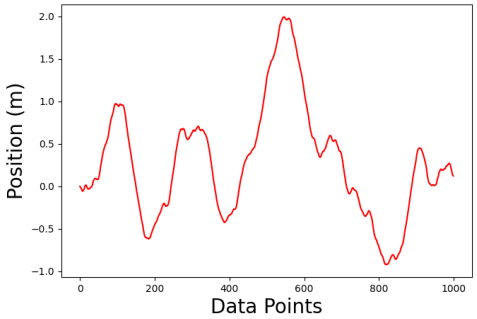


Figure 5: Concentration Gradient Random Walk with Normal Distribution in Second Derivative

This proposed model has the normal distribution in the second derivative of our GPS measurement, with a mean value proportional to our deviation from the true measurement. As the measurement gets further from the true position, it becomes more likely that future readings will turn back towards the true measurement. In addition to this, the model has maximum allowed values set for first and second derivative, to prevent divergence. This model can be seen in Fig. 5.

3.2 Model Parameters

The Noise model can be used with the default parameters, or can have the following parameters set. To get these parameters, it is recommended to get a sample GPS reading from a stationary receiver. The noise model presented above can have the following user specified parameters:

1. Width of distribution; type: positive float. This is the the maximum observed measurement minus the smallest observed measurement of a stationary receiver, in meters.

2. Width of first derivative; type: positive float. This is the maximum observed first derivative of the observed measurement minus the minimum, once again after the distribution was converted to meters.

3. Width of second derivative; type: positive float. This is the maximum observed second derivative of the observed measurement minus the minimum, after converting to meters.

4. Standard deviation of second derivative; type: positive float. This is the computed standard deviation of the second derivative of the observed measurement, once converted to meters.

In addition to this, it is also possible to set just the width of distribution to the desired value, and leave all the other parameters as defaults. This is recommended as this will prevent the model from diverging from the mean. The default parameters provide a standard GPS distribution, which will normally generate a range of about 1 meter deviation on either side of the mean.

4 Software implementation details and API

4.1 Chrono Implementation

This report describes a new noise model implemented in Chrono::Sensor [1] to replace the old model, which followed a standard Gaussian Normal Distribution. For the implementation in Chrono [3], a new noise model class $RandomWalks()$ was added to the Chrono::Sensor module. This is in addition to the original noise model, which remains unchanged. To use the new noise model, a GPS sensor must be created, and the $noise\_model$ parameter must be set to a RandomWalks::Addnoise() object, with parameters as described in section 3. Once the GPS sensor is attached to a rigid body it will generate data for the object.

4.2 ART Implementation

Since ART [2] uses PyChrono [4] as its simulation engine, the structure of Chrono::Sensor::NoiseModel remains the same, but we need to make sure the simulation engine can be built with the $Python\_Module$ enabled. To build PyChrono, it is required to add a compile argument in Chrono::Swig module so that the Chrono::Sensor module can have the $RandomWalks()$ noise model class callable from PyChrono.

5 Numerical Experiments

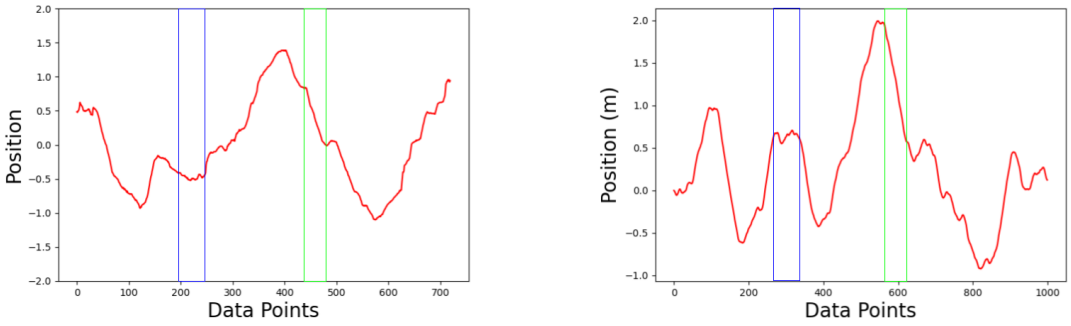
5.1 A qualitative comparison against actual GPS data

In this section, we compare the real GPS data recorded on top of Lot 17 in Madison, WI, to our simulated GPS noise model. We will analyze the results of two experiments. The first experiment consists of the GPS standing still, recording noise at a single location. The second experiment consists of a GPS sensor attached to a moving 1/6th scale vehicle named ART. A GPS sensor in Chrono with the new noise model will be attached to dART, the digital twin of this vehicle, and the same vehicle movement will be performed in simulation [2, 3].

5.1.1 Single point test

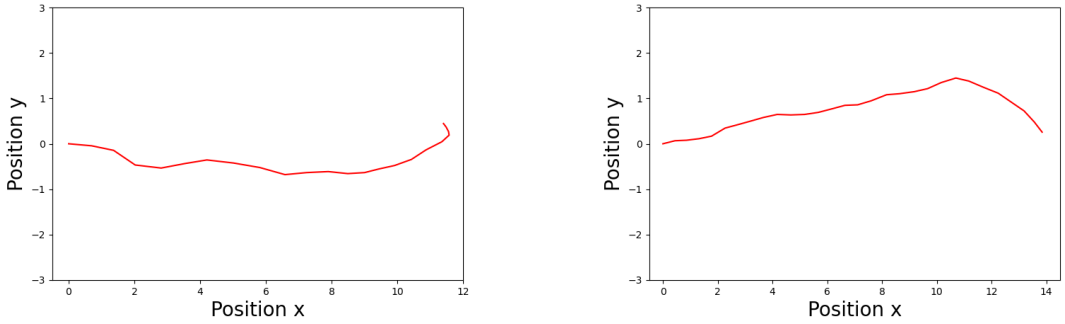
Here we will provide a qualitative comparison of Fig. 1a and Fig. 5. There are two sections of interest: data points 195-245, and 440-480 in Fig. 1a. We will compare these two sections to sections 275-325 and 570-610 of Fig. 5. We can see the highlighted sections in Fig. 6.

From analysis of the second sections (highlighted in green), we see that in both graphs this section is fairly smooth. This appears to occur when the slope of our measurement is fairly large. In contrast, in the first sections (highlighted in blue) we see that while the first derivative of our measurement is small, there appears to be higher observed fluctuation. This makes some sense intuitively based on the noise model, as we have the normal distribution in the second derivative. There will only be small changes in the slope, but these are much more noticeable when we are close to 0.



(a) X Coordinates (b) Simulated Noise Model

Figure 6: Highlighted sections



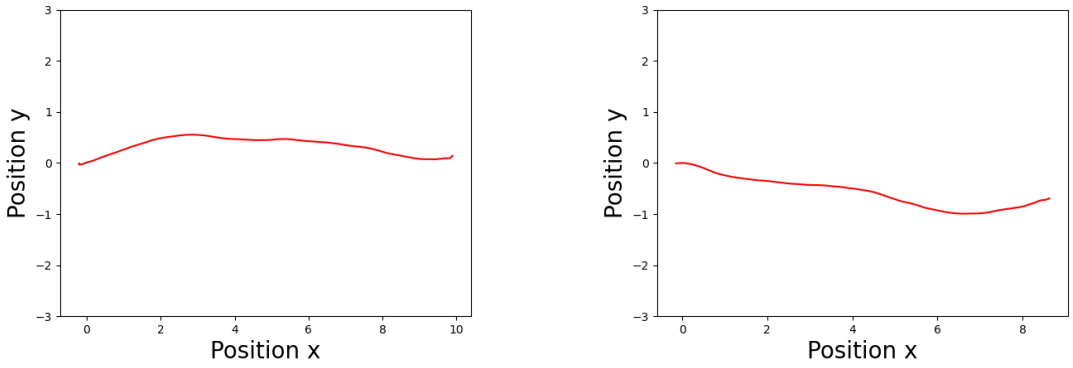
(a) (b)

Figure 7: Real Straight Line Tests

5.1.2 Straight Line Test

The second test consists of driving a scale car with a GPS receiver in a straight line. In this test, a different GPS receiver was used with an update rate of 1 Hz. This demonstrates consistency in the proposed GPS model. In Fig. 7, we show two tests driving ART in a straight line for approximately 20 seconds. This is a demonstration of a moving GPS receiver. In Fig. 8, we demonstrate a simulation running on similar terrain, with the same vehicle input parameters. The GPS noise model is the proposed model above with default parameters. As we can see, the GPS noise model performs similarly to the real GPS data.

Further, a test with the *original* GPS noise model from Chrono is carried out. This noise model uses a normal distribution with a standard deviation of 1m. As can be seen in Fig. 9, this model does not accurately represent the real GPS data.



(a) (b)

Figure 8: Simulated Straight Line Tests

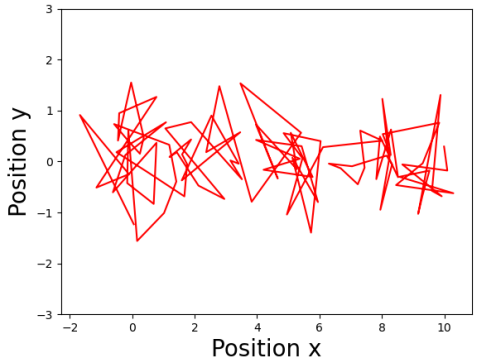
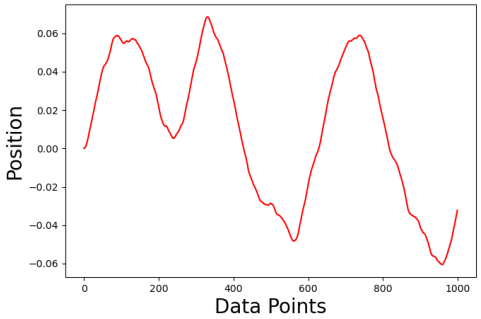


Figure 9: Simulated Straight Line Test with Normal Distribution Noise model

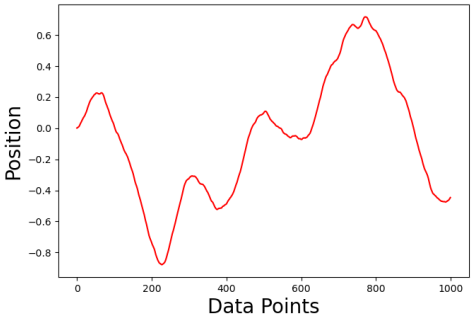
5.2 A study on the influence of the GPS model parameters

There are three parameter setting possibilities recommended: Using the model's default parameters, changing just the width of the distribution, or changing all of the model parameters. The first two options are the "safest," as this will prevent the model from diverging from the mean. With the third option, the user has the most control over the model, and can input their desired parameters described above.

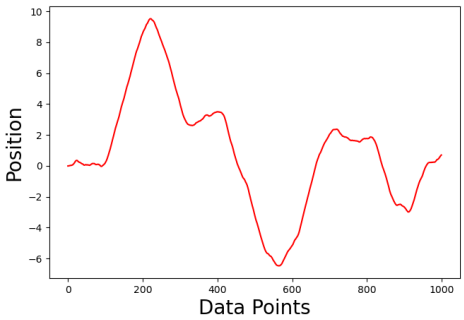
In Fig. 10 we show three cases; a high accuracy demonstration, where the width of the distribution is set to 0.1 m. The second demonstration shows the preset values, with the desired width being 1 m, and the third demonstration shows a low accuracy test, where the desired width is 10 m.



(a) 0.1 m Width



(b) 1 m Width



(c) 10 m Width

Figure 10: Varying Width models

6 Conclusion

This report describes a new GPS model implemented in Chrono::Sensor [1]. Basic tests are provided using the ART/ATK [2] platform to demonstrate this model's capabilities. Next steps include more concrete analysis of this model's performance. An Extended Kalman Filter implementation will be used on the simulated ART vehicle, and its performance will be measured. This algorithm will then be tested in reality to gauge the feasibility of simulation testing for real world applications. Further, an MPC formulation utilizing information from the EKF will be implemented to drive ART long distance through waypoints. This will again be tested in simulation, and compared to reality.

References

[1] A. Elmquist, R. Serban, and D. Negrut, "A sensor simulation framework for training and testing robots and autonomous vehicles," *Journal of Autonomous Vehicles and Systems*, vol. 1, no. 2, p. 021001, 2021.

[2] A. Elmquist, A. Young, T. Hansen, S. Ashokkumar, S. Caldararu, A. Dashora, I. Mahajan, H. Zhang, L. Fang, H. Shen, X. Xu, R. Serban, and D. Negrut, "Art/atk: A research platform for assessing and mitigating the sim-to-real gap in robotics and autonomous vehicle engineering," 2022.

[3] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. Fleischmann, M. Taylor, H. Sugiyama, and D. Negrut, "Chrono: An open source multi-physics dynamics engine," in *High Performance Computing in Science and Engineering – Lecture Notes in Computer Science* (T. Kozubek, ed.), pp. 19–49, Springer International Publishing, 2016.

[4] Project Chrono Development Team, "PyChrono: A Python wrapper for the Chrono multi-physics library." `https://anaconda.org/projectchrono/pychrono`. Accessed: 2023-01-14.