Naval Information Warfare Center PACIFIC

TECHNICAL REPORT 3332
FEBRUARY 2024

Machine Learning Aided Gait Recognition for Inertial Navigation and Orientation – Year 1

Minhdao H. Nguyen
Jeffrey C. Onners
Roger C. Sengphanith
NIWC Pacific

Naval Information Warfare Center (NIWC) Pacific
San Diego, CA 92152-5001

This page is intentionally blank.

TECHNICAL REPORT 3332
FEBRUARY 2024

Machine Learning Aided Gait Recognition for Inertial Navigation and Orientation- Year 1

Minhdao H. Nguyen
Jeffrey C. Onners
Roger C. Sengphanith
NIWC Pacific

Administrative Notes:
This report was approved through the Release of Scientific and Technical Information (RSTI) process in October 2023 and formally published in the Defense Technical Information Center (DTIC) in February 2024.

NIWC Pacific
San Diego, CA 92152-5001

NIWC Pacific
San Diego, California 92152-5001

P.M. McKenna, CAPT, USN
Commanding Officer

M.J. McMillan
Executive Director

ADMINISTRATIVE INFORMATION

The work described in this report was performed by the Non-Linear Dynamics & Materials Branch (Code 71780) of the Basic & Applied Research Division, Naval Information Warfare Center (NIWC) Pacific, San Diego, CA. The Office of Naval Research (ONR) provided funding for this Basic Applied Research project.

Released by
John deGrassie, Division Head
Basic and Applied Research Division

Under authority of
Carly Jackson, Department Head
Cyber/S&T Department

ACKNOWLEDGMENTS

Editor: MRM

EXECUTIVE SUMMARY

OBJECTIVE

This report details the system, test environment, and results used to evaluate a Global Navigation Satellite Systems (GNSS) denied pedestrian inertial navigation system that is aided with velocity estimates from a machine-learning algorithm.

METHODS

A machine-learning algorithm was developed and trained with data from foot-mounted Inertial Measurment Units (IMU) and GNSS data from a user to estimate the user's velocity. After the machine-learning algorithm is trained, the algorithm can estimate the user's velocity with only the foot-mounted IMU data. The velocity estimates are combined with data from a back-mounted IMU and an Extended Kalman Filter (EKF) to estimate the user's position without GNSS data. The system will be evaluated with different terrains and multiple data collections to measure performance across different conditions.

CONCLUSIONS AND RECOMMENDATIONS

The data collections and evaluations described in this report show that the system can estimate the user's position with a range of percent error over distance traveled of 5% to less than 1%. It also shows that the system can work with different terrains and gaits including slow walking, walking, and running.

This page is intentionally blank.

ACRONYMS

CNN	Convolutional Neural Network
ConvBlock	Convolution Blocks
COTS	Commercial Off the Shelf
EKF	Extended Kalman Filter
GLONASS	Global Navigation Satellite System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IMU	Inertial Measurment Unit
MARIO	Machine Learning Gait Recognition for Inertial Navigation and Orientation
MEMS	Micro Electromechanical System
ML	Machine Learning
NavFil	Navigation Filter
PCB	Printed Circuit Board
PPK	Post Processed Kinematic
PPS	Pulse Per Second
RMS	Root Mean Square
UNAVCO	University NAVSTAR Consortium
ZUPT	Zero Velocity Update

This page is intentionally blank.

CONTENTS

FIGURES

TABLES

This page is intentionally blank.

1. INTRODUCTION

1.1 PURPOSE

Global navigation satellite systems (GNSS) are vital for navigation for wide variety of people and institutions, but GNSS has many vulnerabilities. GNSS are easily spoofed or jammed. Also, GNSS can be blocked in indoor and underground environments or tall surrounding structures. It is common for pedestrian navigation to be in areas where GNSS is not accessible. Fortunately, there is much research in GNSS-denied navigation using inertial measurement units (IMU).

1.1.1 Objectives

The main objective of the Machine Learning Gait Recognition for Inertial Navigation and Orientation (MARIO) project is to create a pedestrian navigation system in the absence of GNSS using machine learning and wearable inertial measurement units (IMU). The MARIO system would collect data from the wearable IMUs along with GNSS data to train a machine-learning algorithm to recognize the gait and velocity of a user. Once the machine learning algorithm is properly trained, the algorithm can estimate the user's velocity. The velocity estimate and data from a back-mounted IMU are combined with an Extend Kalman Filter (EKF) to estimate the user's position. The MARIO system would allow for continuous navigation in GNSS-denied environments.

1.1.1.1 Objective 1 – Data Collections

The data from wearable IMUs would vary between different gaits, terrains, and users. The number of variables would be difficult to account for when estimating the user's velocity. Machine-learning algorithms can learn from examples and eventually estimate velocity accurately without much setup. Machine-learning algorithms need to be trained with a good dataset that has a wide variety of data from different users, terrains, and gaits. There will be data collections with wearable IMUs with different terrains and different users. Also, some of the data collections will not be part of the training set. These collections will be used to test and evaluate the MARIO system.

1.1.1.2 Objective 2 – IMU Modules and Ground Truth System

To properly generate the data, wearable IMU modules would have to be developed. Foot-mounted IMU modules and back-mounted IMU modules would be developed. The parts for the modules would be selected for their size and performance specifications. Also, a ground truth system would have to be setup to accurately capture the user's position and velocity.

1.1.1.3 Objective 3 – Machine Learning Velocity Estimator

A machine-learning algorithm would be developed to estimate a user's velocity from the user's wearable IMU data. Freely available machine-learning software would be used and customized. The dataset from the data collections would have to be organized and formatted for the machine learning software to train the algorithm. Once the algorithm is trained, the algorithm can estimate user's velocity from new IMU data.

1.1.1.4 Objective 3 – Extended Kalman Filter Software

The final result of the MARIO system is a position estimate. To calculate the position, an EKF would have to be implemented. The EKF can take the velocity estimates from the machine learning algorithm and data from a back-mounted IMU to estimate a position.

1.2 BACKGROUND

For GNSS-denied pedestrian navigation, dead-reckoning techniques are normally used with micro-electro-mechanical system (MEMS) IMUs since MEMS IMUs are small and light enough to be wearable on a person. Small MEMS IMUs can accurately estimate the orientation of a person, but does not provide accelerators good enough to estimate the position. There has been research in the zero-velocity update algorithm (ZUPT) that uses a foot-mounted IMU to estimate position. When the foot is down and on the ground, the foot is in zero velocity and would stop any error growth while the foot is on the ground [1]. This approach usually requires a tactical grade IMU on the foot and currently tactical grade IMUs are not small enough to be mounted on the foot easily. Also, machine learning has been used with foot-mounted IMU to better estimate when a zero-velocity update occurs and the user's gait [2] [3]. This research usually uses the zero-velocity update algorithm. Our approach would utilize the machine learning research, but apply it in a different manner. In addition, foot-mounted IMUs require wide bandwidth since the foot can experience over 10g of acceleration and over 2000 degrees per second of angular velocity when running. Wide bandwidth IMUs usually higher noise that can increase the error in the position estimation [4].

1.2.1 Inertial Sensors

An inertial sensor measures the acceleration and angular velocity of the object the sensor is mounted on. They come in a wide variety of configurations, performance grades, and technologies. They are used for anything from simple tilt or orientation sensors, to precision-guided munitions, and their price range is anywhere from a few dollars to hundreds of thousands of dollars. Popular types of inertial sensor technologies are based on MEMS, fiber optics, piezoelectric materials and magnetic induction. Inertial sensors are either accelerometers or gyroscopes and typically consist of a combination of the two. Accelerometers are sensors that measure the linear acceleration of the sensor. Gyroscopes measure the angular velocity (rate of rotation) of the sensor [5].

1.2.2 Machine-Learning Algorithm

Machine learning (ML) encompasses a wide variety of algorithms (neural-networks, LSTMs, Reinforcement Learning, Deep Learning, etc.) which describes a computer-aided function generator; the designers aim to make a system that can generate predicted outputs from known inputs, where the function is generated by an algorithmic process applied to datasets [6]. A popular subset of machine learning is Deep Learning, a relatively recent method that emulates a network structure similar to the human brain. Deep Learning includes Convolutional Neural Networks (CNNs), an algorithm that performs convolutions using windows over segments of data to discover common features within a dataset. They are commonly used in image recognition, natural language processing, and time-series forecasting, and have been utilized and successful in previous NIWC Pacific projects [7] [8] [9]. For these reasons, the MARIO project uses CNNs to predict velocity given IMU input.

1.2.3 Extended Kalman Filter (EKF)

IMU data is noisy and cannot be used directly to produce a navigation solution. Kalman Filters can be used to process the data and produce a navigation solution. Kalman Filters can optimally incorporate all the information provided regardless of precision and estimate the values of interest. Kalman Filters require knowledge of the system and measurement devices dynamics, statistical description of noises and measurement errors, and initial conditions of the values of interest. For this case, the values of interest are position, velocity, and orientation. The knowledge of the system and measurement devices dynamics, statistical description of noises and measurement errors is expressed as a linear mathematical model and error estimations [10].

Extended Kalman Filters are a nonlinear version of the standard Kalman Filter. Extended Kalman Filters require more processing, but modern processors are more than fast enough to run in real time [11] [12] [13].

This page is intentionally blank.

2. METHODS

2.1 APPROACH

The main approach for the MARIO system is to create a velocity-aided inertial navigation system where the velocity is estimated using wearable IMUs mounted on the feet. The main IMU for the inertial navigation would be mounted on the lower back. Two IMUs would be on the feet, one on each foot. Figure 1 shows the proposed mounting locations. The velocity is estimated using the acceleration and angular velocity data produced from each IMU and this data is feed into a trained machine learning algorithm that would estimate the user's velocity. Figure 2 shows the relationship between IMU data and velocity.

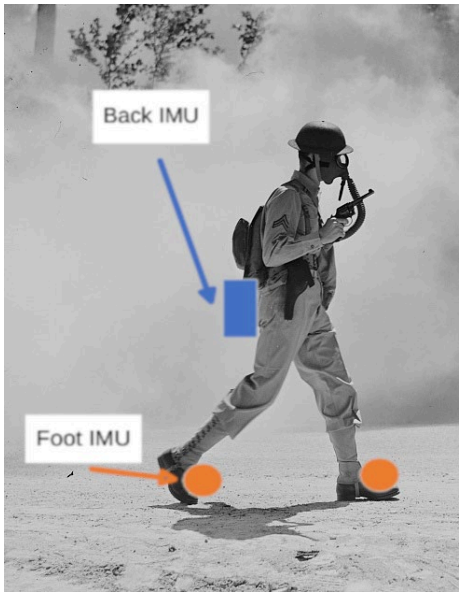

Figure 1. IMU mounting locations.

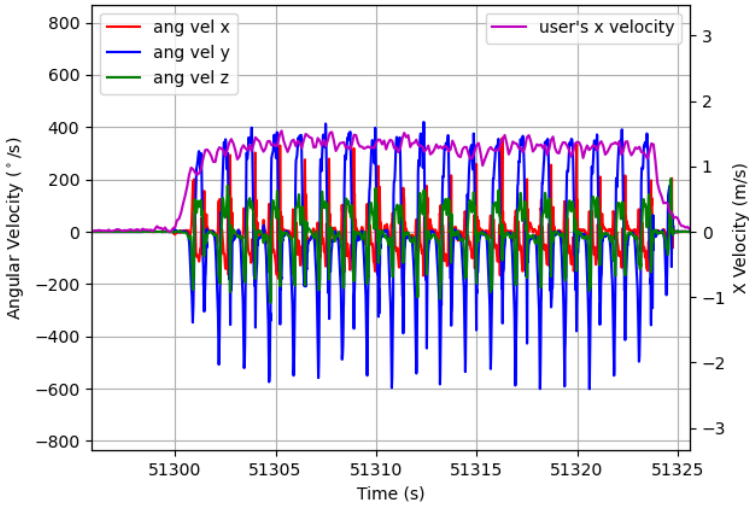

Figure 2. User's IMU data and velocity.

IMU data from the foot and back IMUs and velocity provided by a ground truth system are used to train the machine learning algorithm. The training dataset will be composed of data runs of different users, terrains, and motions (i.e., walking, running, strafing, or walking backwards). There are many differences in the IMU data when there are different users, motions, and other variables even when the velocity may be approximately the same. This wide dataset allows the machine-learning algorithm to train from the variety of data to better estimate the velocity with a wide variety of users and situations. Figure 3 shows the overall data flow of the MARIO system.

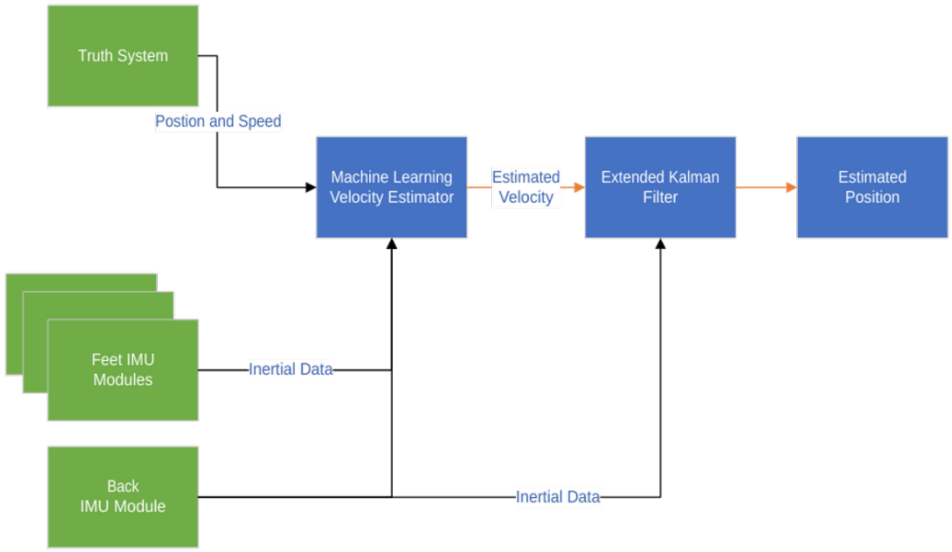

Figure 3. MARIO system diagram.

2.2 SOFTWARE

2.2.1 Machine Learning Model

MARIO uses a Keras-based Convolutional Neural Network (CNN) to predict a continuous velocity value given multi-sensor INS XYZ-axis inputs [14]. A CNN was selected due to the massive amount of support tools available (keras, etc.), its translatability on different datasets, and its success in other NIWC Pacific projects [7] [8].

Modifications to a common CNN structure are necessary for implementation with IMU data. For reference, a generic CNN image classifier could take an image file of size 128 (height pixels) by 128 (width pixels) by 3 (RGB Color channels with 256) to predict a single classification output [15]. These outputs will generally be discrete values and either single-class (true or false representing if a classifiable subject is represented in an image) or multi-class (highest probability selection given a set of possible subjects represented in an image).

A custom CNN was developed to accept raw inertial data as input, and a single float velocity magnitude as output. MARIO's ML model can still be thought of as an image classifier that recognizes pedestrian movement patterns, where the inputs and outputs are more complex. Where an image can have values that range from 0 to 256, IMU sensor data can be negative and its range dependent on the sensor units and resolution. Where a classifier's outputs are usually discrete, velocity is continuous and unbounded. Small errors in velocity predictions can yield poor position estimates, so precise predictions based on minimal available information is crucial. Furthermore, images (usually) represent a single time instance; MARIO deals with time-series data which adds

another layer of complexity. These factors were addressed and represent the contributions we made to the network.

The network is designed in a way to easily test different configurations, mainly with the use of repeatable convolution blocks (ConvBlock) and a Filter-multiplier (M_{Filter}). Each ConvBlock uses the following keras sequence: Conv2D, BatchNormalization, Leaky-Relu activation, low-rate drop-out layer, and MaxPooling2D [14]. Setting the number of ConvBlocks (numblocks) to 4 gave us the best velocity results across users and terrains. We perform and additional ConvBlock (numblocks+1) without pooling, then use 3 Fully Connected Layers to get a single Velocity Output (Out_{vel}). We also use Relu near the end of the network to zero-out any negative weights, since negative velocity predictions are out of the scope for FY23 experiments. Instead of using an activation at the end of our network that would normally bound our possible predictions, we instead use the direct neural network weight output as our velocity prediction. For FY24, the single output value will be changed to two representing an X and Y axis velocity, giving a directional component in addition to magnitude. We currently only use one to represent a person's forward moving speed. We are not using two due to the complexity of obtaining reliable ground truth; a motion capture system may be required to generate X and Y movement. Our final network layers and hyperparameter settings described by Table 1 and Table 2.

Table 1: Layer Descriptions.

Layer	***Data-Shape at Layer***			***Data-Shape at Layer (Actual Values)***
	1st Dim	*2nd Dim*	*3rd Dim*	
Input	A	S	C	(3, 50, 2)
ConvBlock1	A	S	C * M_{Filter}	(3, 25, 32)
ConvBlock2	A	S/2	C * M_{Filter} * 2	(3, 12, 64)
ConvBlock3	A	S/4	C * M_{Filter} * 4	(3, 6, 128)
ConvBlock4	A	S/8	C * M_{Filter} * 8	(3, 3, 256)
ConvBlock5	A	S/8	C * M_{Filter} * 16	(3, 3, 512)
Flatten	A * (S/8) * (C * M_{Filter} * 16)			(2304,)
Dense1	M_{Dense}*2			(512,)
BatchNorm, Relu	M_{Dense}*2			(512,)
Dense2	M_{Dense}			(256,)
Dense3 (Output)	Out_{vel}			(1,)
Legend: Axes (A), Samples (S), Channels (C), Filter Multiplier (F), Dense Multiplier (M_{Dense}), Velocity Outputs (Out_{vel})				

Table 2: Hyperparameters and Variable Values.

Axes (Per Channel)	3
Samples (Per Batch)	50
Channels (Sensors per Batch)	2
Filter Multiplier	16
Dense Outs	256
Velocity Outs	1
Trainable Params	4,065,697
Total Params	4,062,689
Learning Rate	0.001
Epochs	50
Loss Function	Mean Squared Error
Optimizer	Adam
Kernel Size (per Conv2D)	(3, 3)
Dropout Rate (per ConvBlock)	0.1
Dropout Rate (per Dense Layer)	0.4

2.2.1.1 Data-Management

Each sensor (accelerometer, gyroscope) produces three axes of data (XYZ) and can be illustrated as a 3D array. Each array can be stacked along the channel axis, similar to how a PNG image stores RGB color data (Height Pixels * Width Pixels* Color Channels). For IMU data, this would be the number of axes per sensor, times the number of samples per batch, times the number of IMU sensors (Axes * Time Samples* Sensors, or 3 * Seconds/100Hz * 2). Continuous INS data is broken down into windowed time segments. The window length is user-set, and we found half a second (50 samples) per batch to suffice. This is shown in Figure 4.

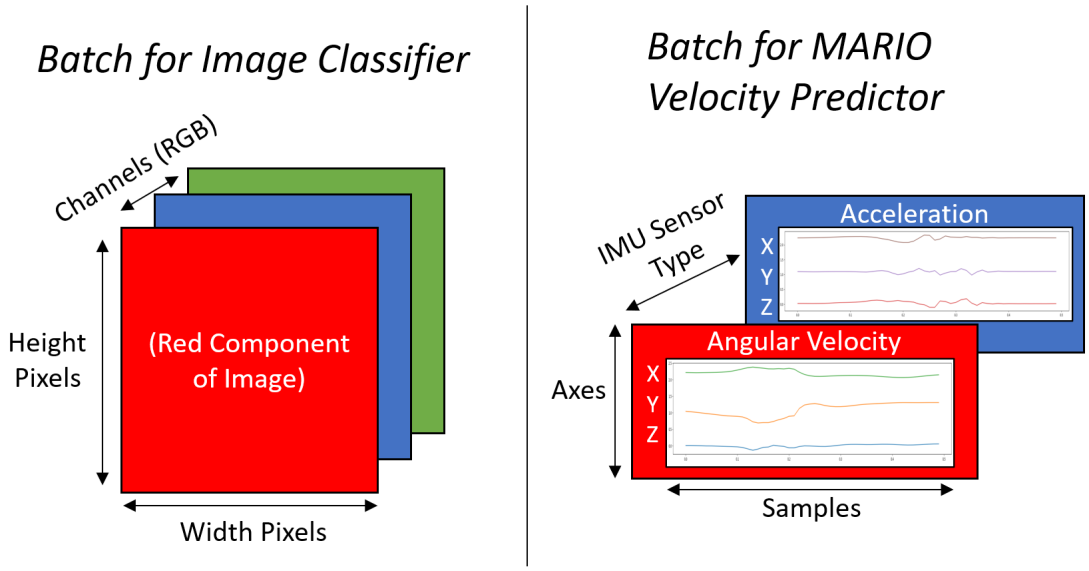

Figure 4. PNG vs IMU batch diagram.

The number of samples the windows shift (S) is variable. Overlap between batches occurs when the S is less than the batch window size as shown in Figure 5 and Figure 6. Including overlap is necessary during testing to simulate a real-time sliding window. During training, including overlap could cause overfitting due to duplicate data in the training dataset [16]. However, movement patterns are quick, causing batches with overlap to appear unique, all while having a similar ground truth. We believe this method can act similarly to transforming an image to help generalize an image classification network. Our final model uses a train shift (S_{train}) value of 25 samples (quarter of a second).

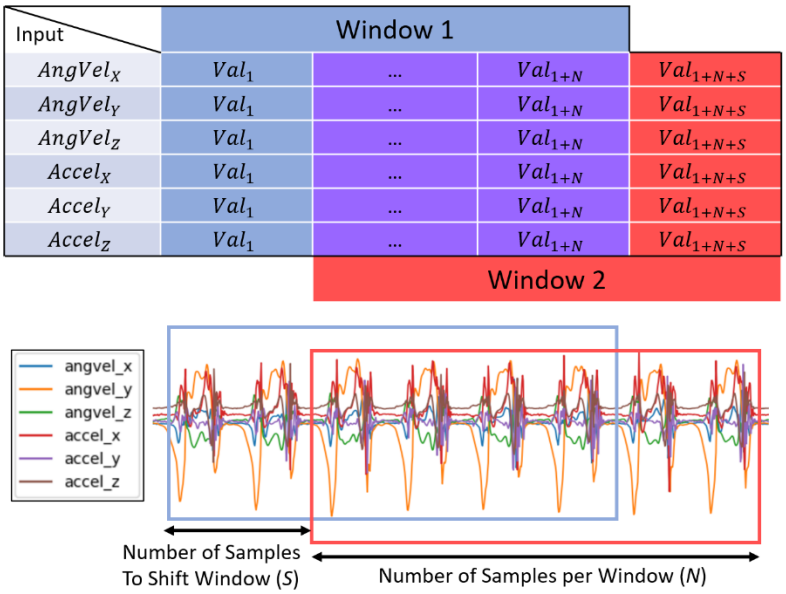

Figure 5. Sliding window diagram.

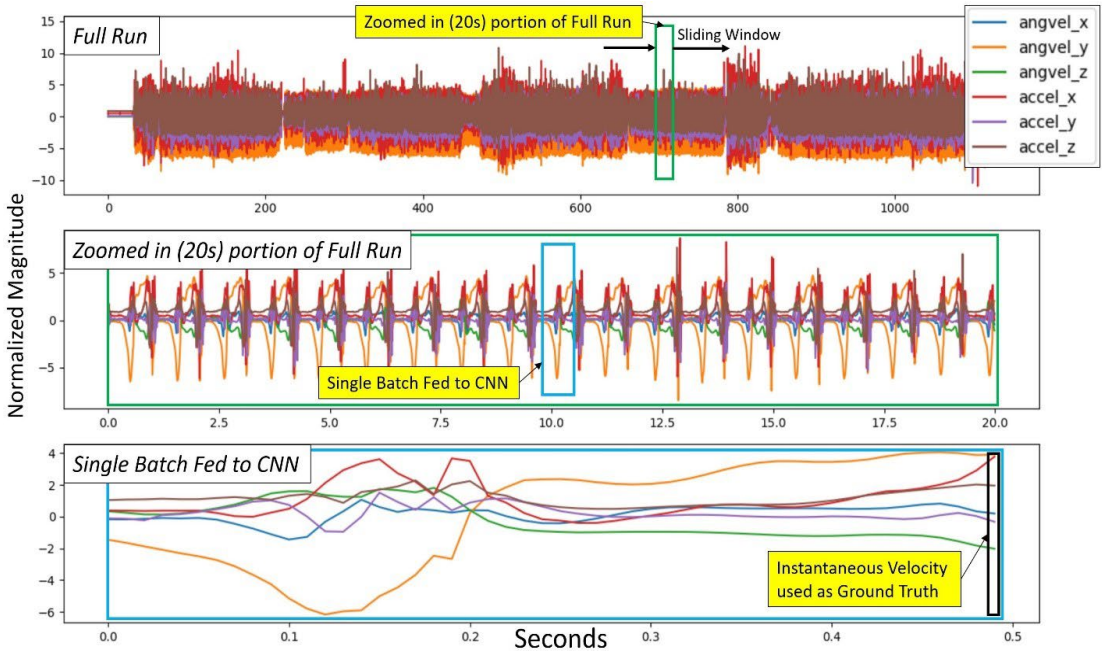

Figure 6. Example of full run and batch (0.5 second window) of IMU data.

During prediction testing, this shift value (S_{test}) becomes whatever number of samples can be handled in real-time. The prediction output frequency is the IMU/Vel frequency divided by S_{test}. A value of 1 sample is equivalent to 100 Hz. If a trained model where to be tested in real time, we would calculate the amount of time required to run a single batch on our online computing device, then choose a value for S_{test} that matches this time. For our purposes, we set S_{test} to 10 samples, equivalent to 10 Hz predictions.

We considered ways to handle a multi-imu system. We decided against using all IMUs together in the same network due to the likely possibility of a single IMU dropout/error causing the whole network to fail. For instance, if we train the network with left foot, right foot, and back IMUs in the same model, but one were to stop recording (low battery, disconnection, etc.), then the inputs for a whole channel would be zero or NaN, causing predictions to be wildly incorrect. For this reason, we decided to train each IMU separately, so each gets its own neural network. Each network makes a velocity prediction, and we use a weighted average to calculate a final velocity as shown in Figure 7. With this methodology, if one sensor is bad, its neural-network predictions will not get included in the velocity average, making the system more reliable.

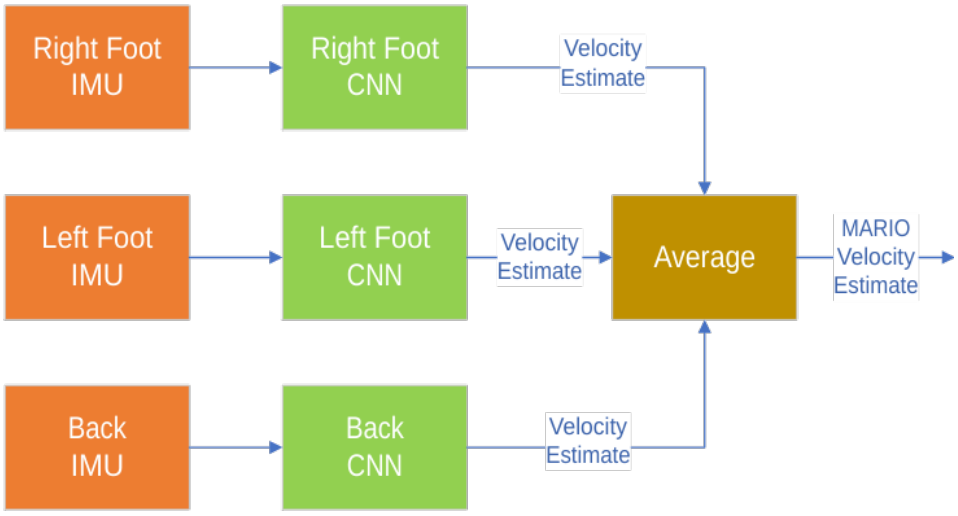

Figure 7. CNN data flow.

2.2.1.2 Scripts

The full process of running the MARIO program is described below:

1) *User Setup*: The user chooses settings for training and testing via command line arguments. This includes CNN parameters like epochs, samples per batch, etc. It also lets the user select which data to train and test on. This is useful for creating experiments with different configurations, such as training on one user/terrain and testing on another.

2) *Obtain and Generate Datasets*: The user-selected data is retrieved and transformed into the format described in Section 2.2.1.1 to be input into the ML model. This process includes syncing all IMU sensor data by timestamp, finding GPS dropouts and replacing missing data with NaN (allowing bad data to be removed from the training set), and interpolating velocity to match the 100Hz IMU data.

3) *Generate Velocity Predictions*: During the training potion, each IMU's data is concatenated, shuffled, and sent through the CNN model. Each IMU's network trains for the user-set number of epochs (default of 50), or until accuracy stops consistently improving. Once the networks are trained, the testing process begins. The test files do not get shuffled and instead stay separated in a list, allowing us to implement a sliding window and reconstruct a full predicted-velocity waveform and compare against the ground truth velocity. We finally send the velocity predictions to the navigation software system in order to get position estimates.

2.2.2 Navigation System

For the navigation solution of the MARIO system, an Extended Kalman Filter called NavFil is used to process the IMU data from the back IMU and the estimated velocity from the machine learning algorithm to produce a position estimate. NavFil was developed at NIWC Pacific. NavFil is given an initial position and initial heading before each run, but does not receive GNSS data. The estimated position is compared with the ground truth data to measure performance.

2.2.3 Ground Truth System

To properly train the machine-learning algorithm, a precise ground truth system is required to provide velocity. For the ground truth of the MARIO system, a global navigation satellite system (GNSS) post-processed kinematic (PPK) was used.

2.2.3.1 GNSS PPK

GNSS PPK uses raw GNSS logs from a GNSS receiver mounted on the user and raw GNSS logs from a nearby base station to produce a position estimate with an accuracy of 2 centimeters. This accuracy allows for a precise ground truth at pedestrian speeds. The base station is stationary with a known location. With that data, corrections can be calculated for each GNSS satellite within view of the base station. These corrections can be applied to the raw GNSS logs from the user's GNSS receiver with an accuracy of 2 centimeters [17].

There are limitations for the GNSS PPK system. The system does not work in real-time and requires post processing after the data is collected. The GNSS receiver needs to be within a 100km from a base station. Tall buildings and trees can block or degrade the GNSS signals so the GNSS PPK estimation may lose accuracy in certain areas. Also, the system cannot work indoors.

2.2.3.2 Base Station Network

The University NAVSTAR Consortium (UNAVCO) maintains a network of GNSS base stations across North America that provides the raw GNSS logs and updates the logs daily [18]. There are several base stations within the San Diego area shown in Figure 8. The main base station used for our data collects is station P475 located on Point Loma.

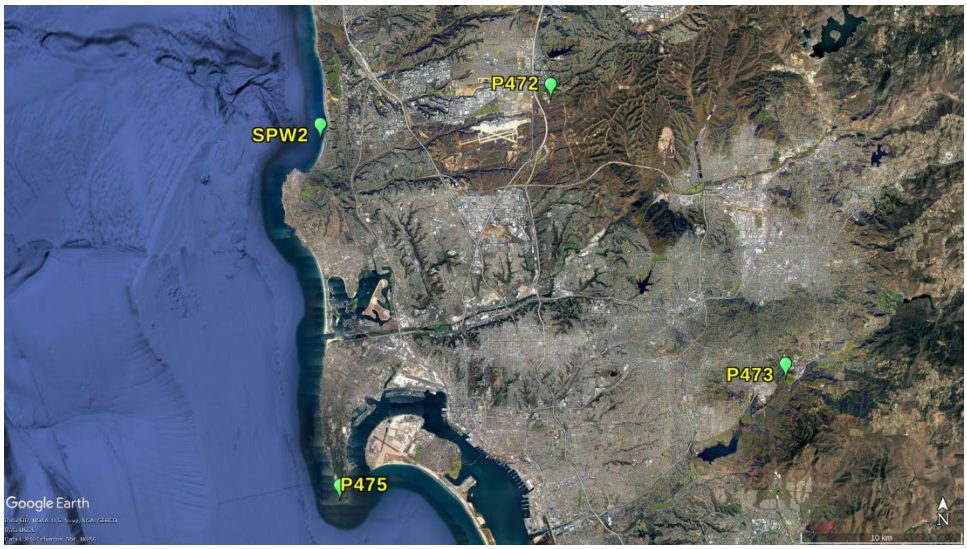

Figure 8. UNAVCO base station map in San Diego.

2.2.3.3 GNSS Receiver

The ublox ZED-F9P is used as the GNSS receiver on the user [19]. The receiver can output raw GNSS logs up to 10 Hz. The GNSS receiver also outputs a pulse per second (PPS) and time message to synchronize data. The data is collected from three different satellite constellations; the Global Positioning System (GPS), the Global Navigation Satellite System (GLONASS) and Galileo satellite system. For the GNSS antenna, a TE Connectivity ANT-GNRM-L12A-3 was mounted on top of a hat with a mounting plate [20]. The antenna was mounted on the hat to allow the least amount of blockage from the view of satellites as shown in Figure 9.

Figure 9. GNSS antenna hat.

2.2.3.4 Post processing software

To generate a PPK position solution, an open-source software called RTKLIB was used [21]. The software uses raw GNSS logs from the base station and the user to produce the position of the user in a latitude, longitude, and altitude format. Also, the software checks the quality of the GNSS signals to estimate the accuracy of the calculated position.

2.2.3.5 Ground Truth Filtering

Due to varying accuracies causing noisy data as shown in Figure 10, further post processing is required for the machine-learning algorithms. A median filter is employed to clean our ground truth data [22]. The median filter takes a sliding window of a set size and takes the median of the window as the output. We use 2% of the input data size as the length of the window. The end of the input is mirrored to preserve the output length. To preserve as much of the original data as possible, a difference between the median filtered data and original data is taken. Differences over a specified threshold are replaced by the filtered values, while values under the threshold are left alone. An example of the filter output is shown in Figure 11.

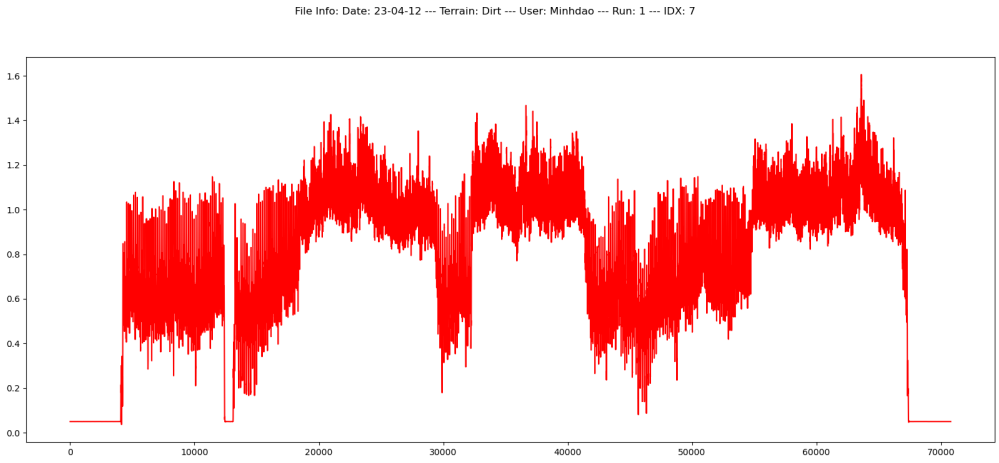

Figure 10. GNSS velocity data before filtering.

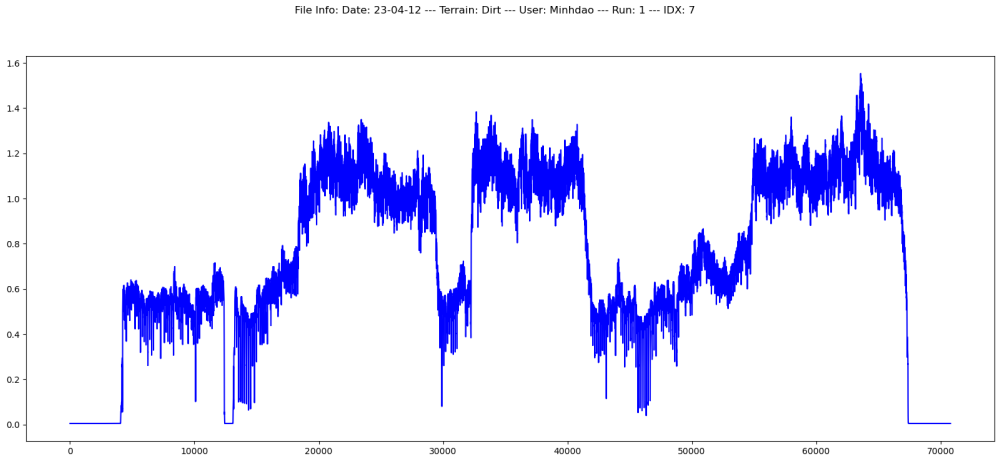

Figure 11. GNSS velocity data after filtering.

2.3 HARDWARE

For the system to work, data had to be collected on several users. Wearable IMU modules were created in-house to better suit the hardware for our purposes. For the first year, the wearables were designed only to collect data. The machine algorithm training and testing and the navigation software are run on a desktop computer in post processing.

2.3.1 IMU Modules

For the first year, two different types of IMU modules were created. One type was created to mount to the feet and the other type to mount to the lower back. Each IMU module has a microcontroller that captures data from an IMU and saves the data to a micro SD card. Also, a GPS receiver is used to synchronize time between each microcontroller. The GNSS receiver outputs a PPS signal and time message with microsecond accuracy. Afterwards, the data is used train or test the machine learning algorithms. The timing is very important to synchronize the IMU data with the ground truth data for proper training of the machine learning algorithm. If the data is out of sync, the algorithm's velocity estimations maybe worse.

In addition, different IMUs were used for the foot IMU modules and the back IMU module. The foot IMU requires a wide range for angular velocity and acceleration, but has higher in-run bias instability. A higher in-run bias instability would poorly affect navigation performance [5]. For the back IMU module, a IMU with a lower in-run bias instability, but lower range for angular velocity and acceleration was selected. The back IMU module experience much lower angular velocity and acceleration than the foot modules so the lower range would suffice. The lower in-run bias instability would allow for better navigation.

2.3.1.1 Foot Module Version 1

The foot module is composed of an IMU, microcontroller, GNSS receiver, battery and a micro SD card. The Raspberry Pi Pico was selected as the microcontroller for the speed, duel cores, and wide support [23]. The IMU in the foot module was the Analog Devices ADIS 16467-3 [24]. This is selected for the wide bandwidth of +/- 2000 degrees per second. This allows for data collects even during running. The in-run bias instability of 6 degrees per hour. The CDtop Technology PA1616D was selected since it had a pulse-per-second (PPS) output that allowed precise timing [25]. All of the hardware is mounted on a custom printed circuit board (PCB) and enclosed in a custom 3D printed enclosure. The enclosure has a mounting mechanism to attach to the laces of a shoe. The foot module is shown in Figure 12.

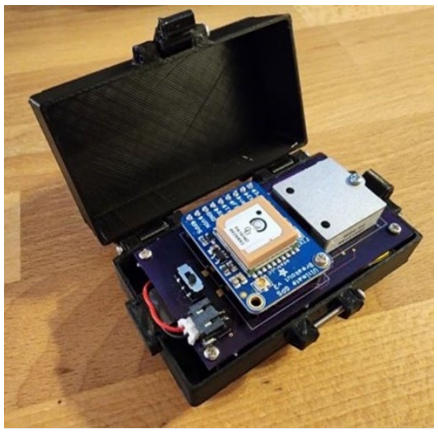

Figure 12. Foot module version 1.

2.3.1.2 Back Module Version 1

The back module version 1 used the same microcontroller, battery, and micro SD card as the foot modules. The selected IMU was the Analog Devices ADIS 16467-2 [24]. This is selected since it was a similar module to foot module IMU so the same software and hardware interface could be used. It has an angular velocity range of +/- 500 degrees per second and an in-run bias instability of 2.5 degrees per hour. The u-blox ZED-F9P was selected as the GNSS receiver since it can output raw GNSS messages that can be used for the GNSS PPK ground truth system [19]. All the hardware is mounted on acrylic plates and a rubber sheet. All this is placed in a running pack that placed around the waist of the user. A GNSS antenna is mounted on a hat connected to the GNSS receiver. The back module version 1 is shown in Figure 13.

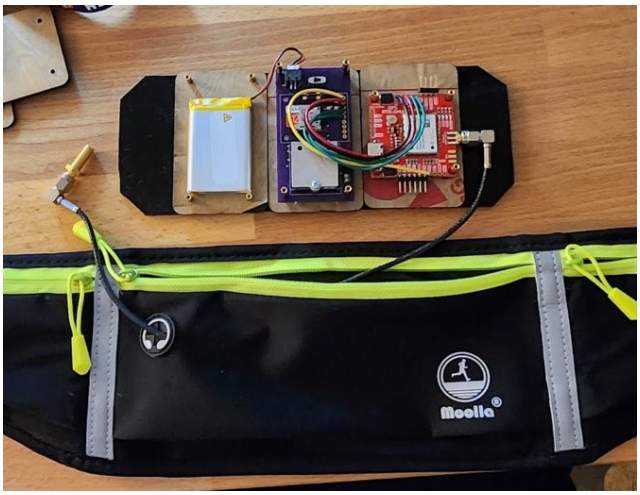

Figure 13. Back module version 1.

2.3.1.3 Back Module Version 2

Later in the year, the back module was redesigned and improved for user-friendliness, improved performance, and simplified hardware setup. The IMU in the back module version 2 is the SBG Systems Pulse-40. It has an angular velocity range of +/- 500 degrees per second and an in-run bias instability of 0.5 degrees per hour [26]. The lower in-run bias instability should improve the navigation performance. The microcontroller, battery, IMU, and micro SD card is mounted in a 3D printed enclosure and mounted on a belt. The GNSS receiver, another microcontroller, and micro SD card is mounted in a separate enclosure that also mounted on the belt with a connector to the IMU enclosure. A power switch is placed and wired towards the front of the belt to easily turn off and on the unit. The unit can be worn around the waist with the IMU enclosure on the lower back. The back module version 2 is shown in Figure 14.

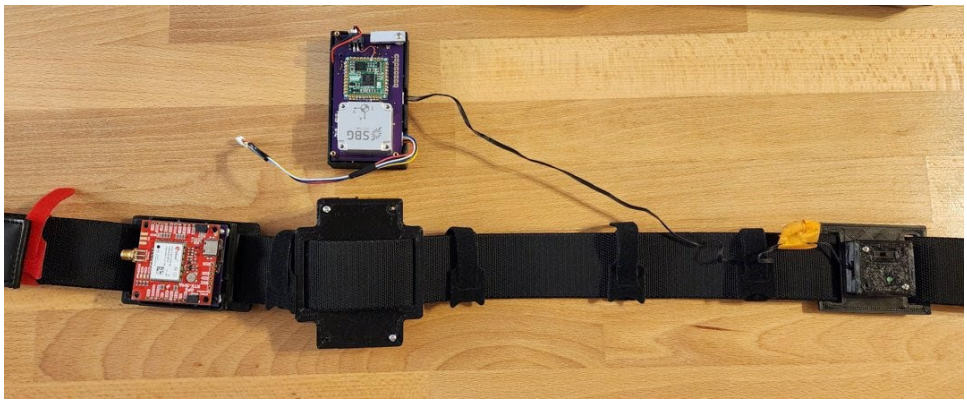

Figure 14. Back module version 2.

2.3.1.4 Back Module Version 3

The back module was redesigned and improved again for user-friendliness and improved performance. The IMU in the back module version 3 is the Microstrain 3DM-CV7-AHRS. It has an angular velocity range of +/- 500 degrees per second and an in-run bias instability of 1.5 degrees per hour [27]. This IMU was selected since it also provided an orientation estimate with relative heading. This allows extra data that improved the navigation performance. The GNSS enclosure and power switch were merged to simplify the unit. The back module version 3 is shown in Figure 15.

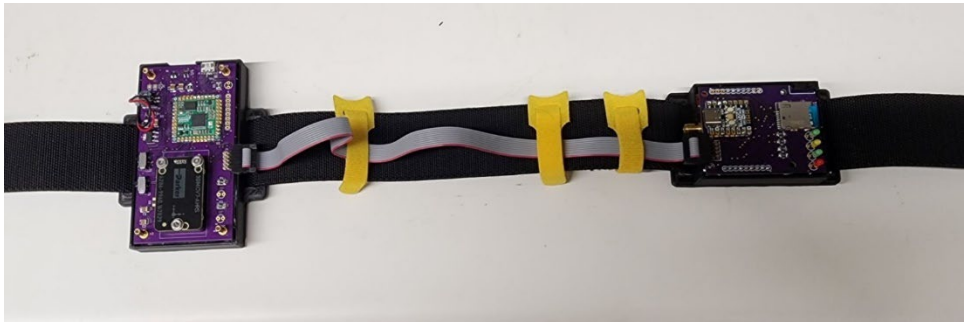

Figure 15. Back module version 3.

2.3.2 Post Processing Computer

The training of the machine learning algorithm is very computationally intense. Once the machine-learning algorithm is trained, the algorithm can run in "testing mode" where the algorithm estimates the user's x velocity. The computer used for our post processing uses the linux operating system and is equipped with multiple NVIDIA Quadro RTX 8000 GPUs.

2.4 DATA COLLECTION

To properly train the machine learning system and test the performance of the entire MARIO system, multiple data collectionss were needed across different variables. The scope of the data was limited to establish a basic initial dataset that the MARIO system can be developed around. After the MARIO system is properly established, more complex data collections can be added the dataset.

2.4.1 Data Collection Method

For each data run, the following rules were followed:

- Only move in the forward facing direction
- Will not move sideways, diagonally, or backwards
- Make slow or quick turns while moving forward
- May turn in place
- Each data run will start with standing still for 30 seconds and then walk straight for about 20 meters to establish initial heading
- Each run may have a different user and different terrain

2.4.2 Data Collection Sites

For the data collection, there were a set of requirements for the hardware and the scope of the project. These are the site requirements:

- Must be outdoors with little obstructions to the sky for the ground truth system, GNSS PPK
- Within 30 km of a GNSS base station
- Site must be a large relatively flat area with mostly one type of terrain
- Low population to allow for free movement during data collection
- Multiple sites will be used with different terrain for each site

2.4.2.1 Site 1 - Fiesta Island

Fiesta Island was chosen as the dirt terrain site. There is a very large section of Fiesta Island that is a flat dirt field with very little obstructions to the sky. The area allows free movement so many different trajectories can be collected. Figure 16 shows an overhead image of the Fiesta Island site and an example route.

Figure 16. Fiesta Island site with example route.

2.4.2.2 Site 2 - South Shore Park

South Shore Park was chosen as the road terrain site. Most of the area is relatively flat and has low amount of obstructions to the sky. A large section of the park is a wide walkway that allows for variation in the trajectory of the data collect. Figure 17 shows an overhead image of the South Shore Park site with an example route.

Figure 17. South Shore Park site with example route.

2.4.2.3 Site 3 - Ocean Beach Athletic Park

Ocean Beach Athletic Park was chosen as the grass terrain site. There is a large section that has flat grass with few obstructions to the sky. Figure 18 shows an overhead image of the Ocean Beach Athletic Park site with an example route.

Figure 18. Ocean Beach Athletic Park site with example route.

This page is intentionally blank.

3. RESULTS

3.1 RESULTS

There were multiple data runs to build a dataset to train the machine-learning algorithm. Six data runs were chosen to use to test the performance of the MARIO system. Two runs on dirt terrain and four runs on road terrain. These runs are not in the dataset that trains the MARIO system. These data runs use the version 3 of the back IMU modules and version 1 of the foot IMU modules. The position, position error, velocity, velocity error, and heading error are plotted to show the overall performance of the MARIO system.

3.1.1 Dirt Terrain Site

The Fiesta Island site was the dirt terrain site. The wide unrestricted area allowed different trajectories for run 1 and 2.

3.1.1.1 Run 1

Figure 19, Figure 20, Figure 21, and Figure 22 shows the results of run 1.

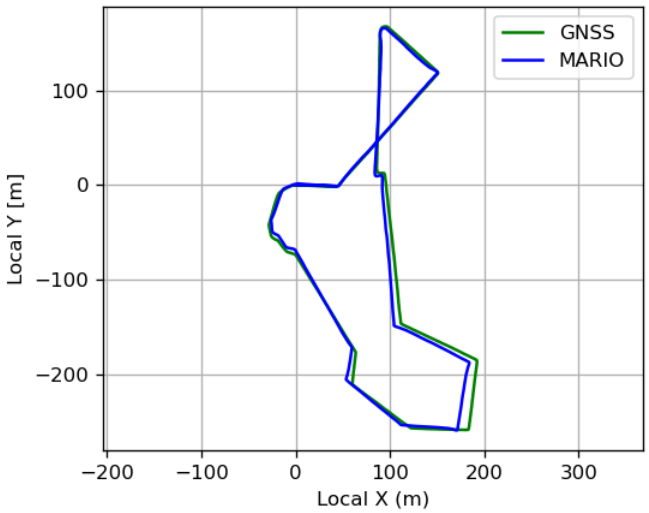

Figure 19. Run 1 local position.

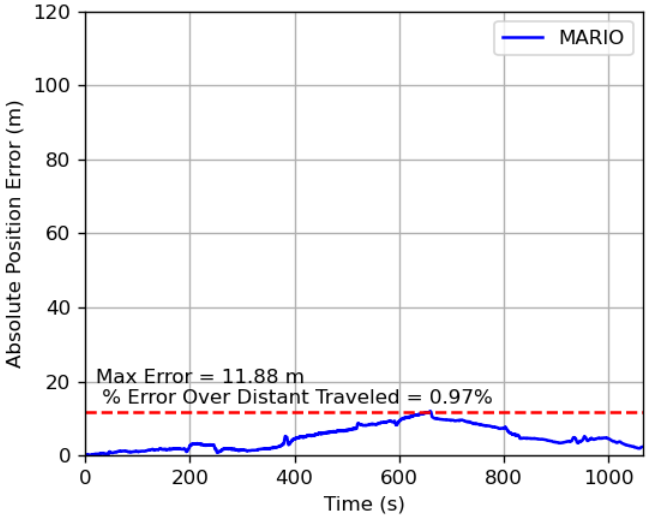

Figure 20. Run 1 absolute position error.

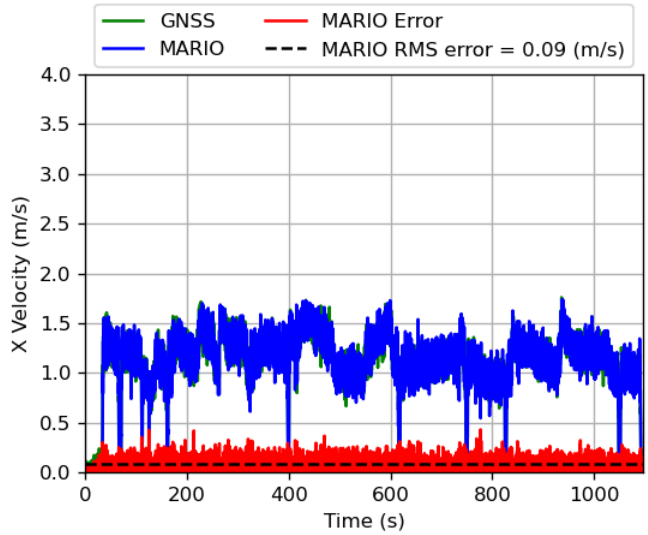

Figure 21. Run 1 X velocity and error.

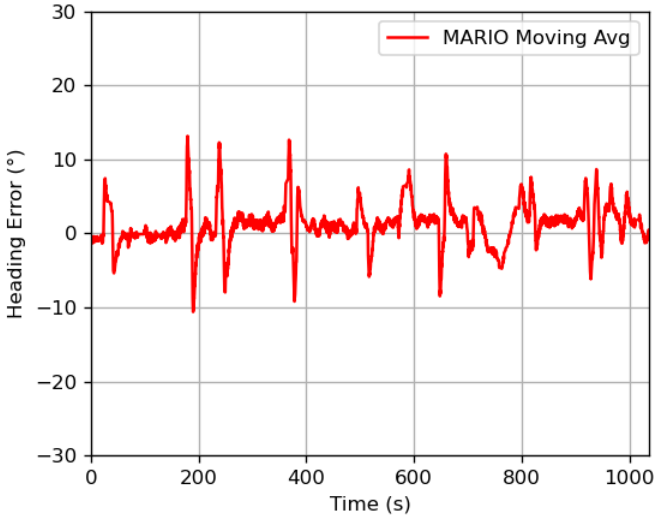

Figure 22. Run1 heading error.

3.1.1.2 Run 2

Figure 23, Figure 24, Figure 25, and Figure 26 show the results of run 2.

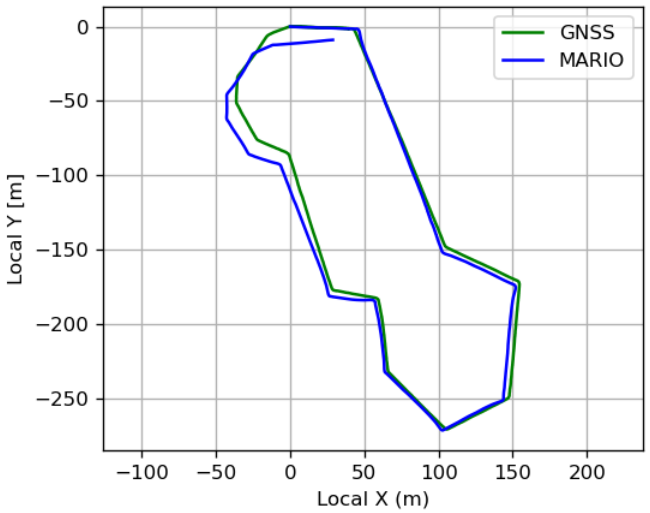

Figure 23. Run 2 local position.

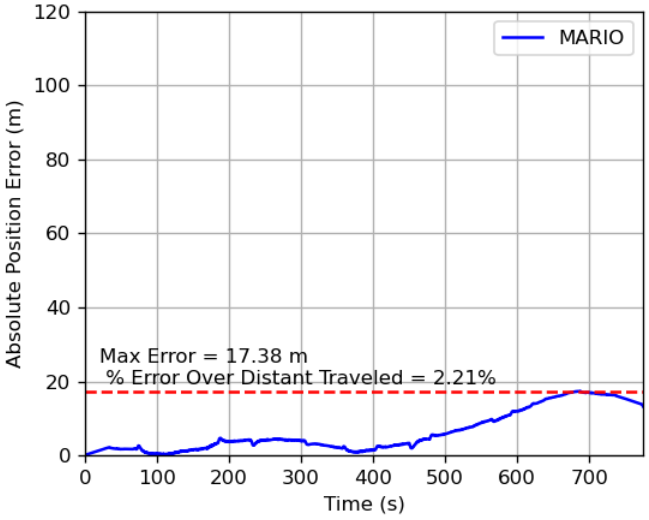

Figure 24. Run 2 absolute position error.

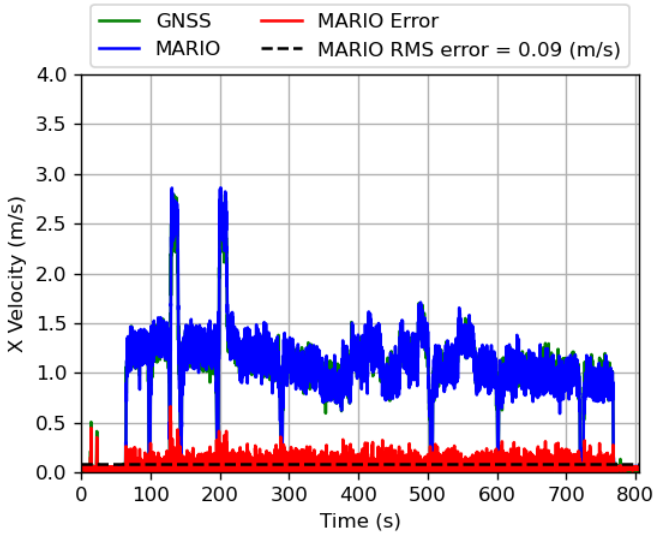

Figure 25. Run 2 X velocity and error.

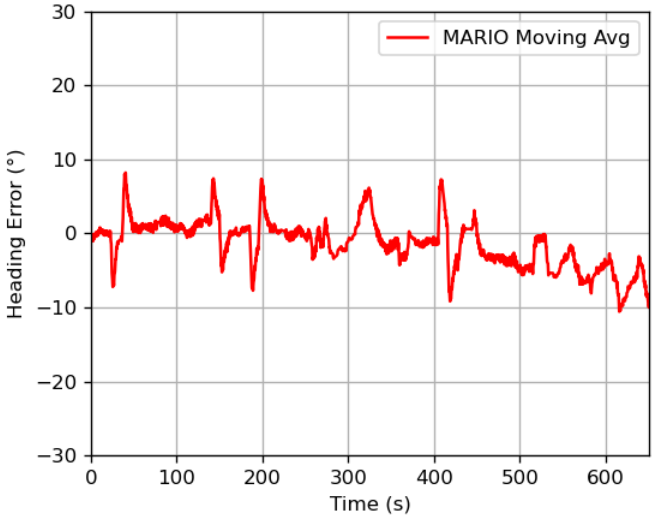

Figure 26. Run 2 heading error.

3.1.2 Road Terrain

South Shore Park was the road terrain site.

3.1.2.1 Run 3

Figure 27, Figure 28, Figure 29, and Figure 30 show the results of run 3.

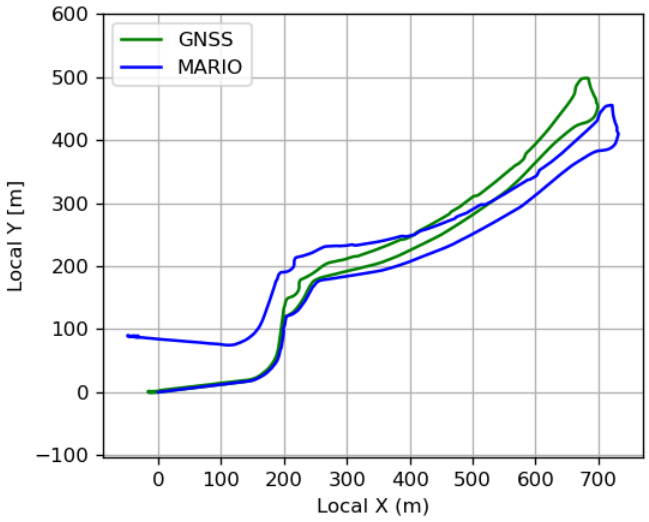

Figure 27. Run 3 local position.

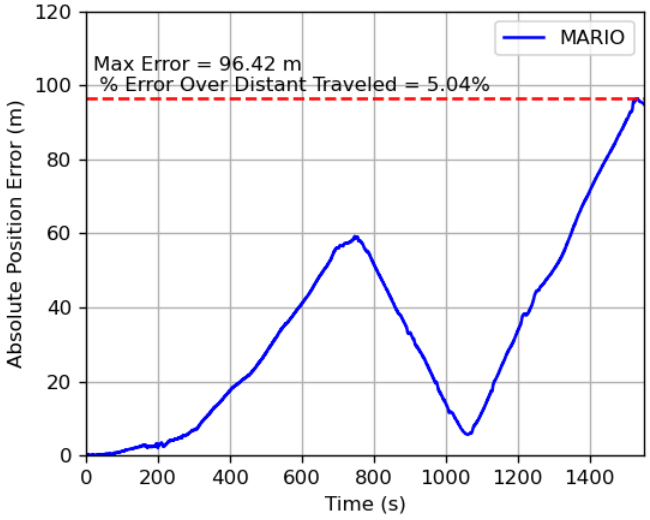

Figure 28. Run 3 absolute position error.

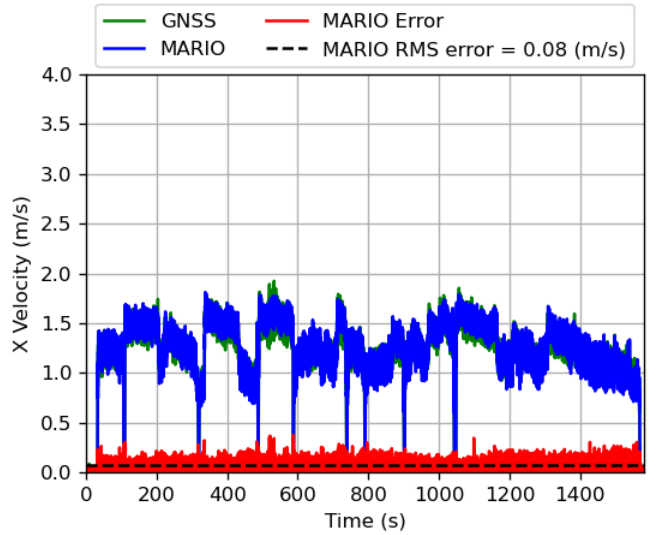

Figure 29. Run 3 X velocity and error.

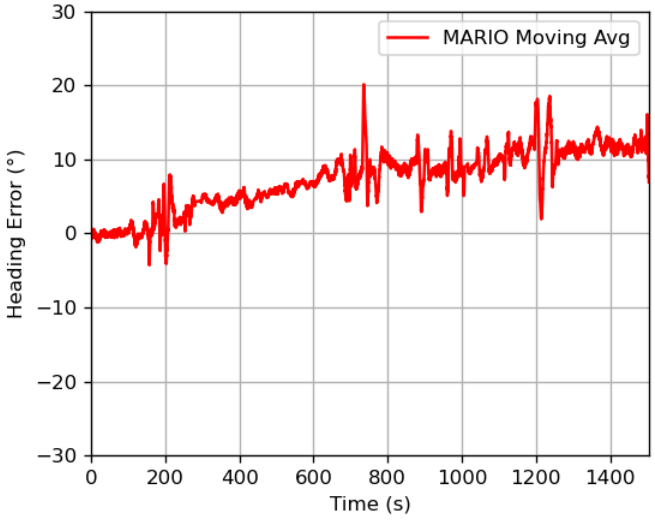

Figure 30. Run 3 heading error.

3.1.2.2 Run 4

Figure 31, Figure 32, Figure 33, and Figure 34 show the results of run 4.

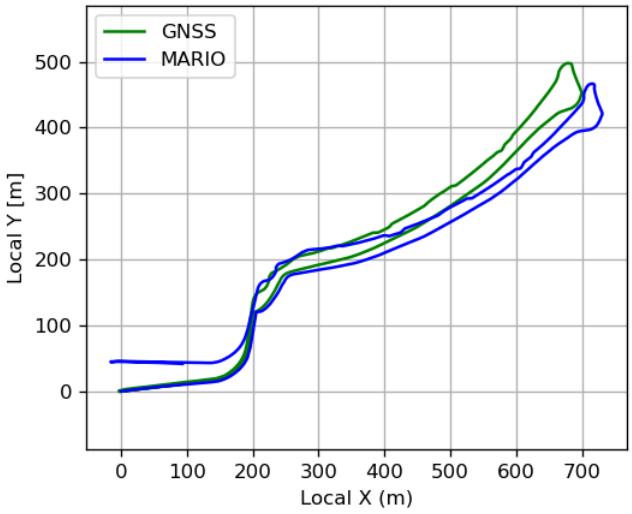

Figure 31. Run 4 local position.

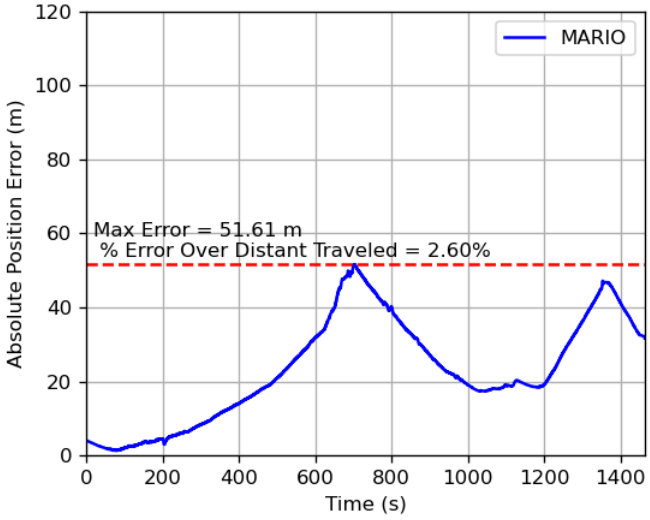

Figure 32. Run 4 absolute position error.

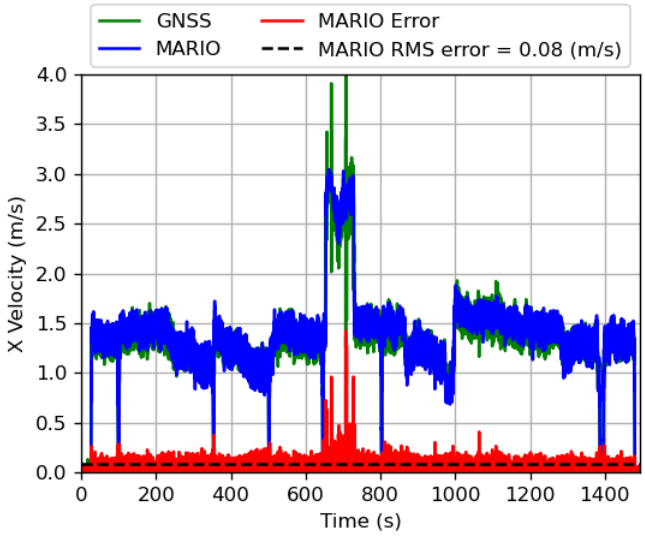

Figure 33. Run 4 X velocity and error.

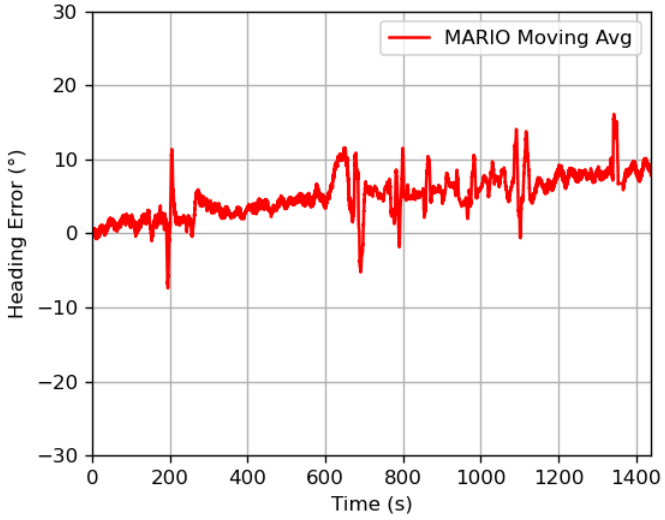

Figure 34. Run 4 heading error.

3.1.2.3 Run 5

Figure 35, Figure 36, Figure 37, and Figure 38 show the results of run 5.

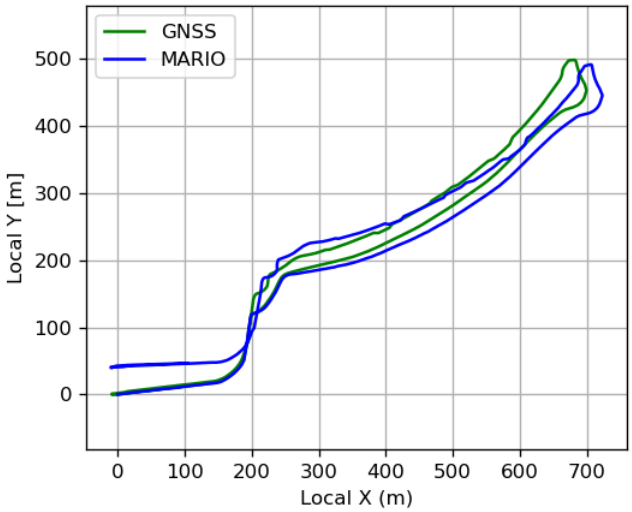

Figure 35. Run 5 local position.

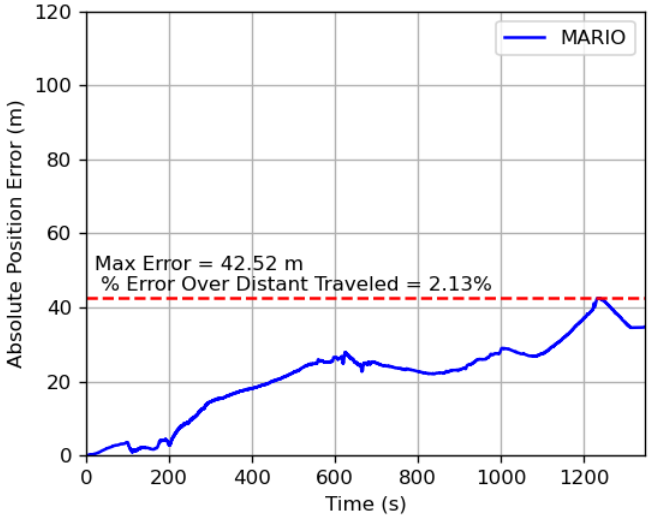

Figure 36. Run 5 absolute position error.

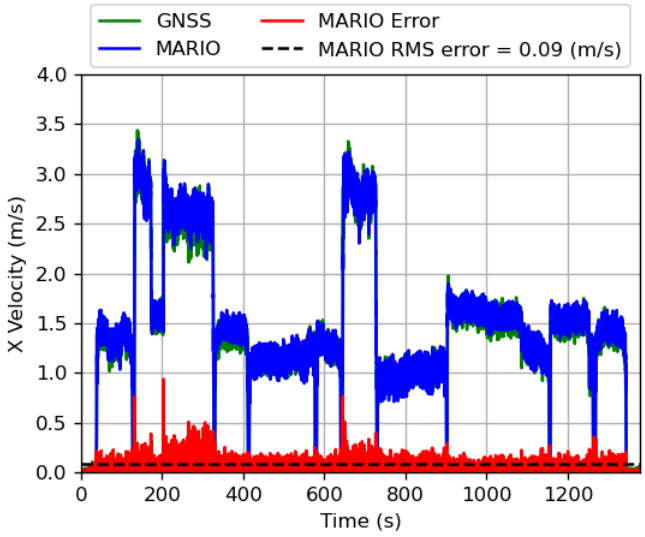

Figure 37. Run 5 X velocity and error.

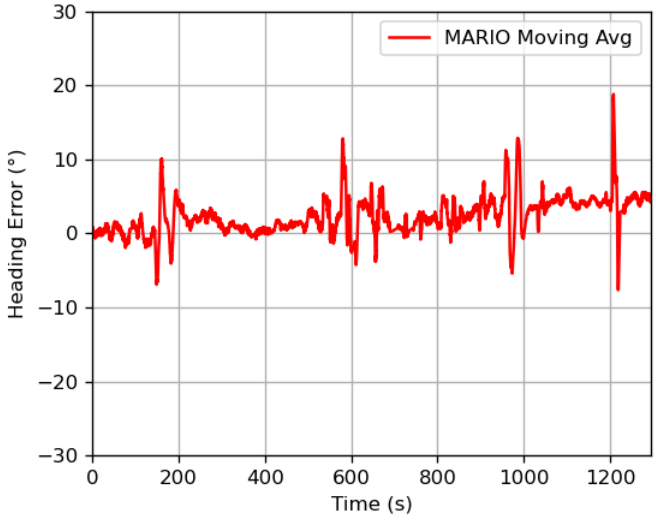

Figure 38. Run 5 heading error.

3.1.2.4 Run 6

Figure 39, Figure 40, Figure 41, and Figure 42 show the results of run 6.

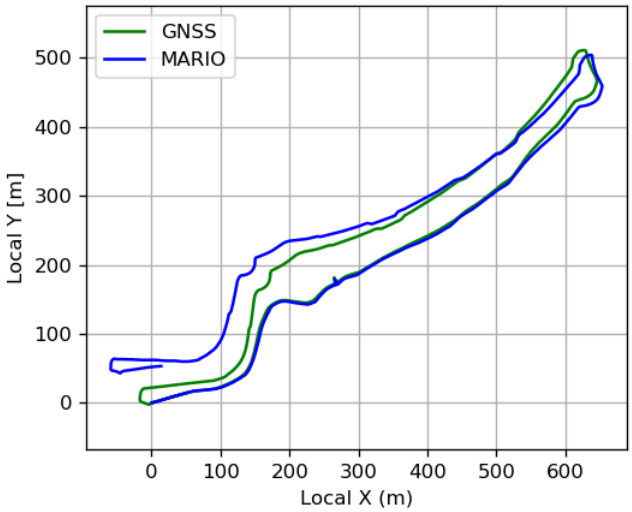

Figure 39. Run 6 local position.

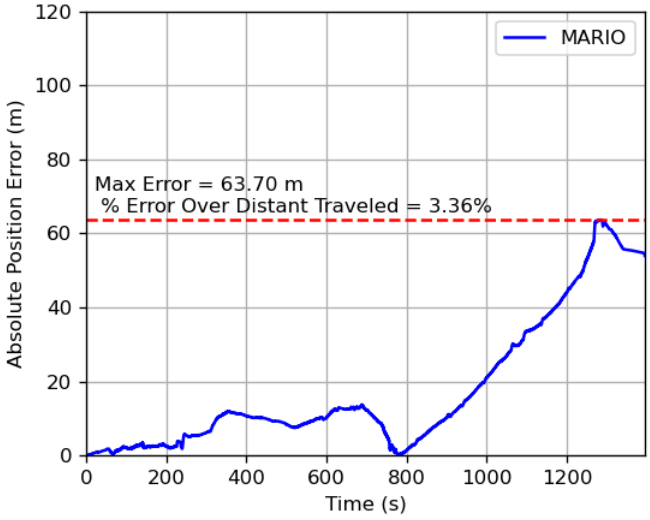

Figure 40. Run 6 absoulte position error.

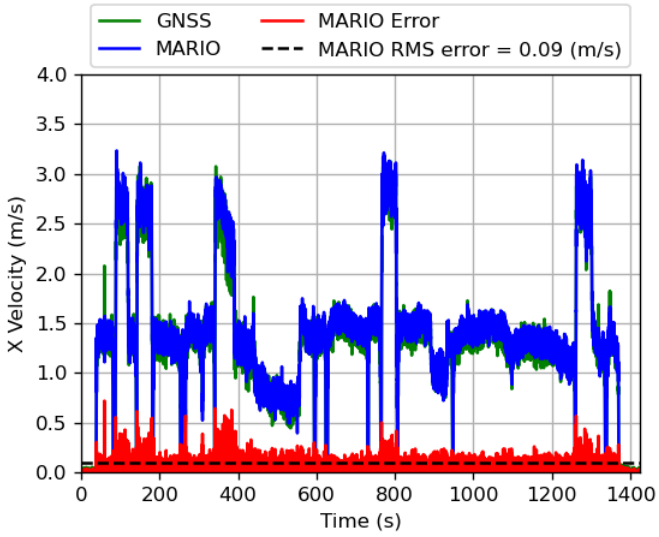

Figure 41. Run 6 X velocity and error.

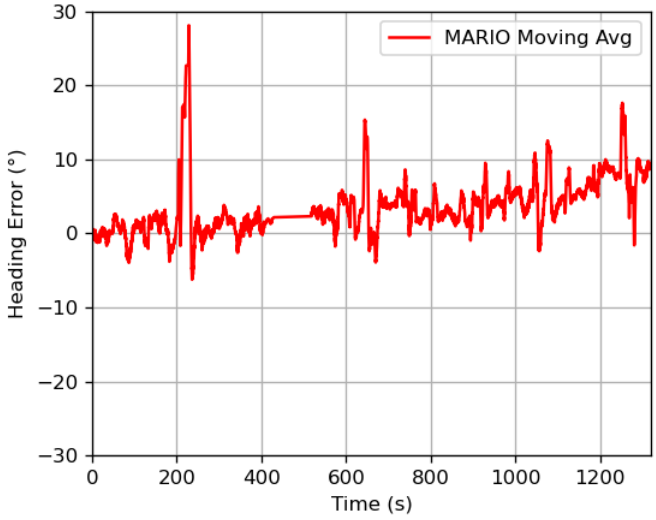

Figure 42. Run 6 heading error.

3.1.3 Key Findings

The key metrics of the six runs are compiled on Table 3.

Table 3. Summary of data run results.

Run	Terrain	Time Length (s)	Total Distance (m)	Velocity X RMS Error (m/s)	Max Position Error (m)	Percent Error over Distance Traveled
1	Dirt	1065	1219	0.09	11.88	0.97
2	Dirt	774	787	0.09	17.38	2.21
3	Road	1550	1913	0.08	96.42	5.04
4	Road	1464	1988	0.08	51.61	2.60
5	Road	1348	1992	0.09	42.52	2.13
6	Road	1395	1895	0.09	63.70	3.36

3.2 STATISTICAL ANALYSIS

Typically, with GNSS-denied inertial navigation, the error grows over the distance traveled. To compare the error between the different runs, the performance metric that we focus on is the percent error over distance traveled. This is the worse position error over the total distance traveled. See equation 1.

$$\%\ error\ over\ distance\ traveled = \frac{(maximum\ absoulte\ error) * (100)}{(total\ distance\ traveled)} \quad (1)$$

To illustrate the performance the machine learning algorithm, the root mean square (RMS) of the velocity x estimate error is calculated for each run. This value shows the average error of the estimate for the entire run. See equation 2.

$$RMS\ Error = \sqrt{\frac{1}{n}({error_1}^2 + {error_2}^2 + \cdots + {error_n}^2)}$$
$$n = number\ of\ values\ in\ entire\ run$$

(2)

The heading error is a noisy signal. To better visualize the heading error, the moving average is calculated and plotted. The moving average of the heading error shows the change over time. For this project, the moving average was calculated from the previous 100 values (previous 10 seconds) from the heading error of interest. See equation 3. This is important to show how the heading solution drifts over time.

$$Moving\ Average = \frac{1}{100} \sum_{i=n-100+1}^{n} heading\ error_i$$

(3)

4. DISCUSSION

4.1 MAJOR FINDINGS

Run 1 had the lowest percent error of 0.97 as shown in Figure 20 and run 3 had the worse percent error of 5.04 as shown in Figure 28. When observing at the position error and heading error of each run, there is a correlation between the two. Run 1 had the lowest amount of heading error shown in Figure 22 and had the lowest percent error. Run 3 had the highest heading error shown in Figure 30 and the highest percent error. The impact of the velocity error does not impact the percent error for each run as shown in Table 3. The heading is an estimation calculated by the lower back IMU integrated Kalman Filter. The magnetometer was disabled since the magnetometer could not be properly calibrated at the time of the data collects.

The velocity error root mean square (RMS) is usually around 0.09 meters per second (m/s) and stays very consistent over different types of terrains and gaits as shown in Table 3. The dirt terrain and road terrain does not have a difference in velocity error RMS. Run 1, 2, 3, and 4 have walking gaits as shown in Figure 21, Figure 25, Figure 29, and Figure 33. Run 5 and 6 have running and walking gaits as shown in Figure 37 and Figure 41. All the runs have several stops during the runs. The velocity error RMS stays consistent with the different gaits. Most of the velocity error likely comes from the noisy velocity from the ground truth system.

In addition, the velocity does not affect the position error. Run 5 has a higher velocity in the beginning of the run shown in Figure 37 and the position error remains low as shown in Figure 36. During this period, the heading error remains low as shown in Figure 38. Run 6 shows similar results. Run 3 was mostly at walking velocity as shown in Figure 29, but had high position error as shown in Figure 28.

Table 3 show that the MARIO system does work with a typical 3% error over distance travelled. The best run had under 1% error. Most of the error comes from the heading error which may be improved with a higher grade IMU for the back IMU module or a magnetometer. The machine learning velocity estimator does work with a low amount of error. Also, the velocity estimation can serve as an input for the EKF.

4.2 FUTURE RESEARCH

There are several subjects of interest for future research. One subject is the implementation of a machine learning algorithm that can estimate the user's velocity in the X and Y direction. This requires a change to the ground truth system that can output the user's position and orientation.

Another subject is to test more expensive and higher performance IMUs to reduce heading error calculations. Ideally, a higher performance IMU with an integrated Kalman Filter is desirable. This allows easier integration into the MARIO system and maximizes the system's performance. There is difficulty to configure a Kalman Filter or Extended Kalman Filter to a particular IMU. An IMU with an integrated Kalman Filter would already be configured and optimized.

In addition, integration of a magnetometer as additional sensor for the system is a subject of interest. Magnetometers can measure the Earth's magnetic field and calculate the heading magnetic north. Unfortunately, magnetometers require calibration and are suspectable to distortions in the magnetic field by metallic objects. This would require research into to properly integrate the magnetometer and check for validity.

In addition, the MARIO system would be better used as entirely mobile embedded system. This means the machine learning algorithm and navigation software have to be implemented in real-time on wearable IMU modules. More advanced hardware would have to be developed to run the software and trained machine learning algorithms.

5. CONCLUSIONS

The MARIO system was used to estimate a user's position using wearable IMUs and machine learning. The best run was under 1% error over distance travelled. Also, the machine learning velocity estimator's velocity error was typically around 0.09 m/s. This was accomplished with relatively inexpensive Commercial off-the-shelf (COTS) components. Also, the velocity error RMS remained consistent between different terrains and different gaits.

The biggest source of position error for the MARIO system is in the heading estimation. Higher performance IMUs or an additional sensor like a magnetometer may aid in reducing the heading error and thus reducing the position error. Further research is required to properly integrate a magnetometer.

The first-year effort of the MARIO project was successful in developing a GNSS-denied pedestrian navigation system using machine learning and wearbale IMUs.

5.1 RECOMMENDATIONS

The main source of error for the MARIO system has been the heading error. The heading drifts over time. If the heading drift can be reduced, then the position error will be reduced and improve the performance of the overall system. A higher performance IMU with an integrated Kalman Filter would help reduce the heading drift with easy integration into the MARIO system.

Also, integrating a magnetometer would aid in reducing the heading drift, but it requires careful calibration and use. Magnetometers measure the Earth's magnetic field and can be susceptible to disturbances in the magnetic field.

A more advanced system to capture ground truth velocity would aid the machine learning part of MARIO. The current GNSS PPK system is very accurate for positioning, but produces relatively noisy velocity. A more accurate system like motion capture would allow for more accurate velocity for a better training dataset. Unfortunately, a motion capture system does have operational area limitations and requires more setup and expense.

This page is intentionally blank.

REFERENCES

[1] E. Foxlin, "Pedestrian tracking with shoe-mounted inertial sensors," *IEEE Computer Graphics and Applications,* vol. 25, no. 6, pp. 38-46, 2005.

[2] Y. Kone, N. Zhu, V. Renaudin and M. Ortiz, "Machine Learning-Based Zero-Velocity Detection for Inertial Pedestrian Navigation," *IEEE Sensors Journal,* vol. 20, no. 20, pp. 12343-12353, 2020.

[3] M. D. Nguyen, "IMU-based Spectrogram Approach with Deep Convolutional Neural Networks for Gait Classification," in *IEEE International Conference on Consumer Electronics*, Las Vegas, NV, USA, 2020.

[4] C. S. Jao and A. M. Shkel, "A Reconstruction Filter for Saturated Accelerometer Signals Due to Insufficient FSR in Foot-Mounted Inertial Navigation System," *IEEE Sensors Journal,* vol. 22, no. 1, pp. 695-706, 2022.

[5] VectorNav, "What is an inertial measurement unit IMU," 3 July 2021. [Online]. Available: https://www.vectornav.com/resources/inertial-navigation-articles/what-is-an-inertial-measurement-unit-imu. [Accessed 30 September 2023].

[6] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, Cambridge, MA: The MIT Press, 2016.

[7] J. Onners, M. Alam, B. Cichy, N. Wymbs and J. Lukos, "U-EEG: A Deep Learning Autoencoder for the Detection of Ocular Artifact in EEG Signal," in *2021 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, Philadelphia, PA, USA, 2021.

[8] M. Jaszewski and S. Parameswaran, "Exploring Efficient and Tunable Convolutional Blind Image Denoising Networks," in *2019 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, Washington, DC, USA, 2019.

[9] E. Bozeman, M. Nguyen, M. Alam and J. Onners, ""Inertial Navigation Compensation with Reinforcement Learning," in *2022 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL)*, Avignon, France, 2022.

[10] P. S. Maybeck, "Chapter 1, Introduction," in *Stochastic Models, Estimation, and Control Volume 1*, Wright-Patterson Air Force Base, OH, Academic Press, 1979, pp. 1-16.

[11] S. Boyd, "Lecture 9 The Extended Kalman Filter," Stanford University, Stanford, CA, USA, 2008.

[12] Mathworks, "Extended Kalman Filter," Mathworks, [Online]. Available: https://www.mathworks.com/help/control/ref/ekf_block.htm. [Accessed 30 September 2023].

[13] G. A. Einicke and L. B. White, "Robust extended Kalman filtering," *IEEE Transactions on Signal Processing,* vol. 47, no. 9, pp. 2596-2599, 1999.

[14] Keras Special Interest Group, "About Keras," 28 April 2020. [Online]. Available: https://keras.io/about. [Accessed 30 September 2023].

[15] J. McDermott, "Convolutional Neural Networks - Image Classification w. Keras," 2023. [Online]. Available: https://www.learndatasci.com/tutorials/convolutional-neural-networks-image-classification. [Accessed 30 September 2023].

[16] C. Shorten and T. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data,* vol. 6, 2019.

[17] United States Geological Survey, "USGS Global Positioning Application and Practice," 6 April 2017. [Online]. Available: https://water.usgs.gov/osw/gps. [Accessed 30 September 2023].

[18] EarthScope Consortium, "All Real-time Networks & Stations Monitoring," 23 May 2014. [Online]. Available: https://www.unavco.org/instrumentation/networks/status/all/realtime. [Accessed 30 September 2023].

[19] U-blox, "ZED-F9P Integration Manual," 31 August 2023. [Online]. Available: https://content.u-blox.com/sites/default/files/ZED-F9P_IntegrationManual_UBX-18010802.pdf. [Accessed 30 September 2023].

[20] TE Connectivity, "ANT-GNRM-L12A Series L1/L2 Magnetic Mount Active GNSS Antennas," October 2022. [Online]. Available: https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FANT-GNRM-L12A_Series%7FA%7Fpdf%7FEnglish%7FENG_DS_ANT-GNRM-L12A_Series_A.pdf%7FANT-GNRM-L12A-3. [Accessed 30 September 2023].

[21] RTKLIB, "RTKLIB: An Open Source Program Package for GNSS Positioning," 2020. [Online]. Available: https://www.rtklib.com/. [Accessed 30 September 2023].

[22] SciPy community, "The main SciPy namespace," 2023. [Online]. Available: https://docs.scipy.org/doc/scipy-1.11.3/reference/generated/scipy.ndimage.median_filter.html. [Accessed 30 September 2023].

[23] Raspberry Pi Foundation, "Raspberry Pi Pico," [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-pico. [Accessed 30 September 2023].

[24] Analog Devices, "Precision MEMS IMU Module ADIS16467," February 2020. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/adis16467.pdf. [Accessed 30 September 2023].

[25] CDtop Technology, "PA1616D," [Online]. Available: https://www.cdtop-tech.com/products/pa1616d. [Accessed 30 September 2023].

[26] SBG Systems, "User manual," 10 February 2022. [Online]. Available: Available: https://support.sbg-systems.com/sc/imu/latest/user-manual. [Accessed 30 September 2023].

[27] LORD MicroStrain, "MicroStrain 3DM-CV7 Series," 2023. [Online]. Available: https://www.microstrain.com/sites/default/files/8400-0141_3DM_CV7_Datasheet.pdf. [Accessed 30 September 2023].

This page is intentionally blank.

INITIAL DISTRIBUTION

84310	Technical Library/Archives	(1)
71780	M. Nguyen	(1)
71740	J. Onners	(1)
71740	R.Sengphanith	(1)

Defense Technical Information Center
Fort Belvoir, VA 22060–6218 (1)

Office of Naval Research (1)

This page is intentionally blank.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-01-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
February 2024	Final	

4. TITLE AND SUBTITLE

Machine Learning Aided Gait Recognition for Inertial Navigation and Orientation- Year 1

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHORS

Minhdao H, Nguyen
Jeffrey C. Onners
Roger C. Sengphanith
NIWC Pacific

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NIWC Pacific
53560 Hull Street
San Diego, CA 92152–5001

8. PERFORMING ORGANIZATION REPORT NUMBER

TR-3332

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
875 N. Randolph Street
Arlington, VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)

ONR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release. Distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report details the system, test environment, and results used to evaluate a Global Navigation Satellite Systems (GNSS) denied pedestrian inertial navigation system that is aided with velocity estimates from a machine learning algorithm. A machine learning algorithm was developed and trained with data from foot-mounted Inertial Measurment Units (IMU) and GNSS data from a user to estimate the user's velocity. After the machine learning algorithm is trained, the algorithm can estimate the user's velocity with only the foot-mounted IMU data. The velocity estimates are combined with data from a back-mounted IMU and an Extended Kalman Filter (EKF) to estimate the user's position without GNSS data. The system will be evaluated with different terrains and multiple data collects to measure performance across different conditions. The data collections and evaluations described in this report show that the system can estimate the user's position with a range of percent error over distance traveled of 5% to less than 1%. It also shows that the system can work with different terrains and gaits including slow walking, walking, and running.

15. SUBJECT TERMS

MARIO; Machine Learning; IMU; Inertial: GNSS; GPS: Pedestrian

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Minhdao H. Nguyen
U	U	U	SAR	60	19b. TELEPHONE NUMBER *(Include area code)* (619) 226-5248

Standard Form 298 (Rev.
Prescribed by ANSI Std. Z39.18

This page is intentionally blank.

This page is intentionally blank.

Approved for public release. Distribution is unlimited.

Naval Information Warfare Center (NIWC) Pacific
San Diego, CA 92152-5001