CEVA MotionEngine™

Navigating the Complexities of Robotic Mapping White Paper

Rev. 1.0

October 2021

Documentation Control

History Table

Version	Date	Description
1.0	07 October 2021	Initial revision

Disclaimer and Proprietary Information Notice

The information contained in this document does not represent a commitment on any part by CEVA®, Inc., or its subsidiaries (collectively, "CEVA"). CEVA makes no warranty of any kind with regard to this material, including, but not limited to implied warranties of merchantability and fitness for a particular purpose whether arising out of law, custom, conduct or otherwise.

Additionally, CEVA assumes no responsibility for any errors or omissions contained herein, and assumes no liability for special, direct, indirect or consequential damage, losses, costs, charges, claims, demands, fees or expenses, of any nature or kind, which are incurred in connection with the furnishing, performance, or use of this material.

Support

CEVA® makes great efforts to provide a user-friendly software and hardware development environment. Along with this, CEVA provides comprehensive documentation, enabling users to learn and develop applications on their own. Due to the complexities involved in the development of DSP applications that might be beyond the scope of the documentation, an online Technical Support Service has been established. This service includes useful tips and provides fast and efficient help, assisting users to quickly resolve development problems.

How to Get Technical Support:

- **FAQs**: Visit our website http://www.ceva-dsp.com or your company's protected page on the CEVA website for the latest answers to frequently asked questions.
- **Application Notes**: Visit our website http://www.ceva-dsp.com or your company's protected page on the CEVA website for the latest application notes.
- **Email**: Use the CEVA central support email address ceva-support@ceva-dsp.com. Your email will be forwarded automatically to the relevant support engineers and tools developers who will provide you with the most professional support to help you resolve any problem.
- **License Keys**: Refer any license key requests or problems to sdtkeys@ceva-dsp.com. For SDT license keys installation information, see the *SDT Installation and Licensing Scheme Guide*.

Email: ceva-support@ceva-dsp.com

Visit us at: www.ceva-dsp.com

Table of Contents

1. INTRODUCTION 1

1.1 Scope 1

1.2 Audience 1

2. SOURCING THE RIGHT SENSORS 3

2.1 Dead Reckoning 3

2.2 Wheel Encoder Characteristics 4

2.3 IMU Sensor Characteristics 5

2.4 Optical Flow Sensor Characteristics 6

2.5 Dead Reckoning Sensor Fusion 7

2.6 Characterizing Characteristics 8

3. TRACKING SUCCESS 10

3.1 Measuring Up to Specifications 10

3.2 Life Imitates… the Test Environment 11

3.3 Expecting the Unexpected 11

4. ANALYSIS AND CONCLUSIONS 13

4.1 Reading the Lines 13

4.2 Conclusion 15

5. GLOSSARY 17

Figure 2-1 Dead Reckoning Error 4
Figure 2-2 Wheel encoder components 4
Figure 2-3 Different surfaces can change the effective wheel circumference 5
Figure 2-4 Angular error (θ) and corresponding positional error (δ) 6
Figure 2-5 Optical flow tracking 7
Figure 2-6 Example variations in ZRO - Temperature relationship 9
Figure 3-1 Robot motion capture 10
Figure 3-2 Robot vacuum test room 11
Figure 4-1 CDF of multiple algorithms over different conditions 13
Figure 4-2 Trajectory error metrics 14
Figure 4-3 Relative error distribution for a variety of fusion methods and sensor conditions 15

List of Tables

Table 3-1: Acronyms 17

1. Introduction

Robotic systems are a complex beast, requiring the combination of carefully picked mechanical, electrical, and programmed parts.

Before a line of code is written, you have to pore over spec sheets to determine which mechanical components stand a chance of achieving your end goal. Every robot also requires sensors to understand the world around them, then rely on mapping algorithms to move where we want them to, not just where the code tells them (old programming joke).

After you have something that proves your concept, you need to test more to fine tune the idea over time. Testing is another sophisticated and arduous process, to measure and analyze what the robot is seeing relative to what you see and want.

If you're tired of reading this already, that's understandable. Even with this high-level summary of the process, there are still so many combinations of places where choices need to be made and things can go wrong.

1.1 Scope

This paper aims to highlight issues you should think about when designing a sensor system for your robot and discuss how you can gain confidence in the process. Specifically, it will include sensor qualification and sensor choice, key performance metrics, creating a test plan, setting up a testing environment, and an example analysis of the data. While we will explain the process in the context of our work with robot vacuums, the principles apply to any ground-roving robots.

1.2 Audience

If you are jumping into wheeled robot design, but aren’t sure where to start, this document can help! If you have some experience already and just want to check if you’re missing anything, this document can help you too.

2. Sourcing the Right Sensors

When selecting the right parts for your robot, choosing sensors is a pivotal step. The data that goes into the robot's decision-making determines what it sees. If that data misrepresents the real world, the robot is destined to behave undesirably. Garbage in, garbage out.

Let's consider the sensing needs of a wheeled robot vacuum. The first thing to know is that there are three broad classes of navigation systems:

1. **Random Walk** – these robots are going to use proximity sensors and wall bumper sensors to bounce around their environment in a mostly random fashion. They will cover a given area eventually, but must stop cleaning with enough energy left to find their way back to the charger or risk getting stranded with a dead battery. How do we get back to that charging station again? Inexpensive but inefficient.
2. **Intelligent Walk** – these robots aim to improve the user experience and improve battery life, by making smart use of some combination of wheel encoders, Inertial Measurement Units (IMUs), and optical flow sensors. Robots that can clean in a smarter pattern and find their way back to that charger in the other room will do a better job and keep their users happier.
3. **Full Mapping (SLAM)** – SLAM robots add a LIDAR sensor or wide-angle camera to map their environment and navigate in very complex areas. These sensor components add to the hardware cost while also requiring a more powerful processor to handle the data. These robots may still fall back on intelligent-walk strategies to handle dark or featureless areas.

Clearly the IMU, wheel encoder, and optical flow sensors can be useful in both Intelligent Walk and SLAM systems. Let's take a closer look at what these can do and how you can optimize them.

2.1 Dead Reckoning

The key to all Intelligent Walk navigation techniques is **Dead Reckoning**. Sounds scary, but it's really just the process of estimating the robot's location relative to a starting position using measurements of speed and direction over time. The fundamental problem here is that the estimated position will inevitably diverge from the true position due to small measurement and estimation errors that add up over time. Because dead reckoning doesn't use any absolute reference points, minimizing the rate of this position error growth is very important. Let's review the sensors involved and find out where these errors come from.

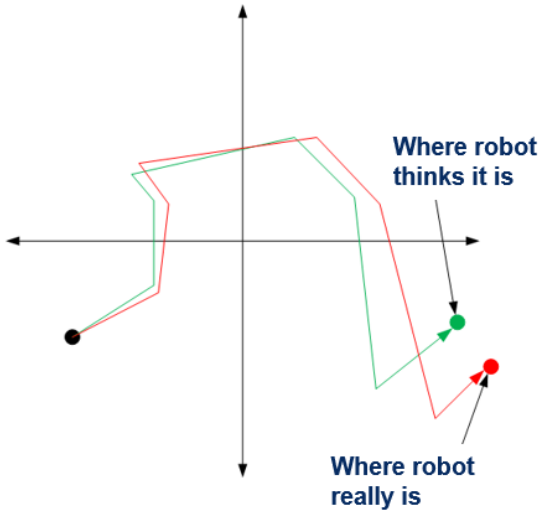

Figure 2-1 Dead Reckoning Error

2.2 Wheel Encoder Characteristics

Wheel encoders use optical or magnetic mechanisms to measure the rotation of your robot's wheels (forward and reverse). These sensors register a "tick" for each fraction of the wheel rotation, and the ticks calibrate to a known distance around the circumference of the wheel, corresponding to a distance traveled along the floor. The finer your tick resolution, the more precisely you can measure the wheel distance.

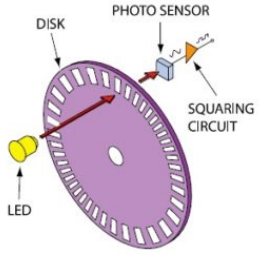

Figure 2-2 Wheel encoder components

This sounds great, but there's a catch. Robot wheels slip and skid, sometimes in obvious ways like if the robot gets stuck on an obstacle, and sometimes even when you don't notice it. On some surfaces like dirty floors or thicker carpet, the wheels are continuously slipping a little with every bit of rotation. Imagine walking up a sand dune!

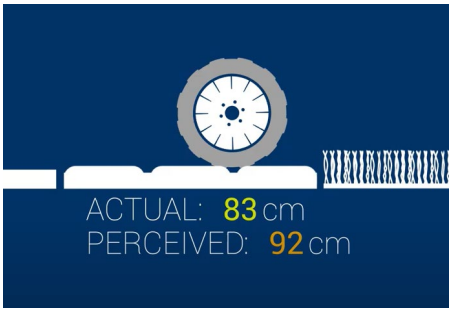

Figure 2-3 Different surfaces can change the effective wheel circumference

Robots may cross a variety of surfaces while doing their job, so being able to adapt to the conditions will help to improve overall accuracy.

2.3 IMU Sensor Characteristics

Maintaining accurate heading is the key component to following a desired path. You can calculate heading from the wheel encoders, but due to the slips we already mentioned, this is not always the most accurate. A better source is the Inertial Measurement Unit (IMU).

The IMU itself comprises an accelerometer and gyroscope, and sometimes an additional magnetometer. They have the key components to provide a digital sense of inertial motion and heading. Accelerometers track the force of linear acceleration, and when there is no movement, the direction of gravity. This helps provide a long-term measure of orientation. Gyroscopes track angular velocity and produce short-term orientation measurements. Magnetometers measure the magnetic field around them and provide long-term orientation based on this field.

Let's take a look at some characteristics to consider when comparing inertial sensors.

- Gyroscope Bias/Zero Rate Offset (ZRO)
- Accelerometer Bias/Zero Gravity Offset (ZGO)
- Gyroscope scale
- Vibration effects
- Sensor noise
- Sensor rate and quantization
- Temperature hysteresis
- Axis performance differences

Of these, the most influential on sensor accuracy are the gyroscope scale, gyroscope bias, and accelerometer bias.

Gyroscope scale is a multiplicative error in the gyroscope output relative to its motion. For instance, if there was a scale error of 1%, then the gyroscope will be a degree off for every 100 degrees traveled (101° or 99°). This impacts any ground-roving robot by influencing the heading that it believes it's traveling by that same amount. Cancelling rotations helps minimize error. A cancelling rotation is simply rotating the opposite of what you did before (e.g. turning clockwise a *perceived* +180 degrees and turning back a *perceived* -180 degrees). The more uncancelled rotations a robot makes, the more the scale error is multiplied.

Gyroscope bias, or Zero Rate Offset, is the angular rate seen while the sensor is at rest (ideally 0). But since angular position is determined as the integral of angular rate, any offset error leads to an ever-growing heading error. Any angular error is extended along the robot's path, creating a scaling positional error. As seen in Figure 2-4, the robot's angular error (θ) contributes to its growing positional error (δ).

*Figure 2-4 Angular error (**θ**) and corresponding positional error (**δ**)*

Accelerometer bias, or Zero Gravity Offset, is a similar principle to ZRO. It is the latent acceleration reading while the device is still (ideally equal to gravity). Accelerometer output offsets result in pitch and roll errors. These offsets can affect overall orientation calculations and, in the case of a magnetometer, tilt compensation.

In theory linear acceleration can be used to calculate velocity (integration) and position (double-integral), but even small inaccuracies from bias and noise can cause the error in integrated velocity and position values to grow exponentially over only a short time period.

2.4 Optical Flow Sensor Characteristics

An optical flow sensor tracks the movement of the floor below the robot to provide a 2D estimate of velocity. This is the same technology in a computer mouse, but on a larger scale: it illuminates the floor, detects tiny features on the surface, and measures their movement between frames.

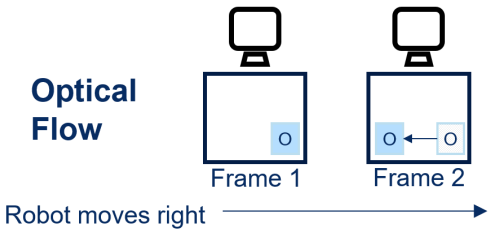

Figure 2-5 Optical flow tracking

The advantage of looking at the floor is that the sensor can track robot movement independent of any wheel slippage. On the other hand, the perceived motion of a feature is a function of both velocity and distance between floor and sensor (consider how slowly a telephone pole on the horizon moves when compared to a telephone pole along the side of the road). If this distance is not precisely calibrated, or varies during use, measurement accuracy will suffer. For example, if the surface is closer to the sensor than expected (e.g. moving from tile to carpet), the measured velocity may be higher than the true velocity.

Optical flow measurements can also be affected by the floor surface type. Some very smooth or dark floors may not have enough features for tracking, and then the measurements can be completely wrong. Some optical flow sensors have multiple illumination modes, e.g. LED or laser, with each mode having its own strengths and weaknesses. It makes sense to use the type that performs best for the surface you're on, but you have to know when to switch.

2.5 Dead Reckoning Sensor Fusion

Clearly there are situations where each of these sensors works well, and when performance of one or more might break down. Sensor fusion is the process of choosing which sensors' data to rely on at each moment, and how to combine their information for a more accurate overall state estimate.

For example, we can compare the velocity estimates derived from each sensor in order to self-calibrate the scale of optical flow measurements while in operation. This eliminates the need for factory calibration and allows the system to adapt to changing operating conditions.

Once the optical flow is calibrated, we can continue to use filtered measurements from each sensor to estimate the quality of data from optical flow and wheel encoders. These quality estimates are critical for determining when to trust data from sensors that are performing well and when to reject data from sensors that aren't.

2.6 Characterizing Characteristics

Now that we understand the core concepts, we need to select the right sensors. One strategy is to choose sensors that fall within your desired tolerances based on the listed mins and maxes in the datasheet. But it would be ideal to have a comprehensive look at the sensor characteristics and map out these anomalies to better understand how to work with or around them.

The best way to account for the negative sensor characteristics is to study them with data! An ideal system for this would require a number of features to be helpful for sensor analysis. It would require a large amount of data collection and unique permutations of position, operating mode, temperature, and more to characterize the sensor qualities we've discussed. This automated data collection system would require:

- A statistically significant number of sensors
- Custom PCBs to communicate with these sensors
- Equipment to iterate through permutations of
    - Position
    - Operating mode
    - Temperature
    - Humidity
    - Magnetic field
- Data collection and analysis tools
- External storage to log the data

The permutations are arguably the trickiest part of this system to setup. Single axis rotations can be stepped through with a precision gimbal. Putting that single-axis gimbal perpendicular to another allows full tri-axis motion. Operating mode can be changed through communication from each PCBs processor. Put these boards and gimbals in a temperature-controlled chamber, and you can modify temperature at the same time. An environmental chamber can artificially age these boards with humidity, and a Helmholtz coil can be used to change magnetic fields. Add in a computer to run these systems, log the data, and the software to analyze it, and you have a full sensor characterization system.

While the system is complex to setup, the results provide insights that are well beyond what any datasheet says. For instance, if we revisit the ZRO characteristic, you can see its relationship with temperature over all the sensors tested.

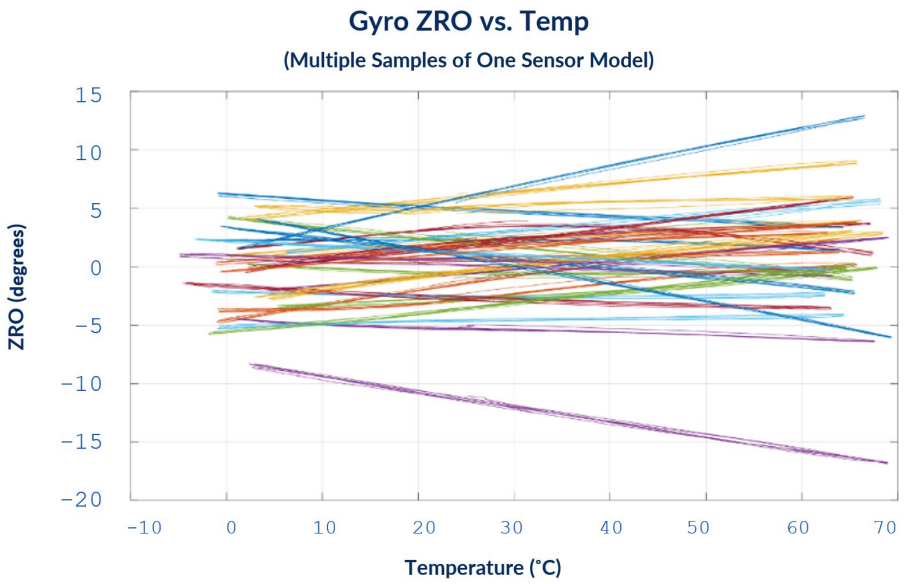

Figure 2-6 Example variations in ZRO - Temperature relationship

Each line in the data above depicts the effect of changing temperature on ZRO for a single sensor (all sensors are the same model). Each sensor behaves differently as temperature changes. Some are strongly positive, others negative, some are less influenced by temperature. But all of them have different ranges of effect.

The same level of detailed data collection can be done with the other parameters (offset, gain, skew, rotation, noise, non-linearity) with varying conditions of temperature, voltage, aging, and mode. Some of these effects can be modeled and compensated, and some reflect limitations that dictate the performance limits of your system. One must carefully weigh sensor characteristics against external factors (like cost and integration effort) to make the best selection for your robot application.

3. Tracking Success

After sensors are picked and the system is put together, testing is necessary to prove out navigation algorithms and ensure consistent high performance. But how do you measure performance?

In the case of a ground-roving robot, its localization algorithm needs to accurately track its location, while other algorithms help it achieve its larger function. The algorithms designed for navigation and those fulfilling a robot's objective are meaningless without proper direction.

This is especially true of a cleaning robot that needs to cover an entire surface to finish its job. This principle applies to all land-roving robots, not just vacuums. The more accurate its mapping, the faster it finishes its job, and the happier the end user. A hospitality robot can lead visitors to the proper location quickly and without incident. Accurate movement from a large warehouses' robots means that customers are getting their products that much faster, and at scale.

Since wheeled robots tend to move in straight lines, heading accuracy and heading drift are strong metrics to improve. Heading is only part of the equation, however. Heading is a component of where the robot is going, but where it ends up is the most important. Measuring trajectory error will help us understand how accurate our state estimation really is.

3.1 Measuring Up to Specifications

We've decided on measuring heading and trajectory error. Great! But now we need a truth to compare to our robot's outputs. In the case of tracking motion, we've found flexibility, accuracy, and precision using an IR-based camera system. It's the same technology used for movie motion capture and in robotics labs around the world.

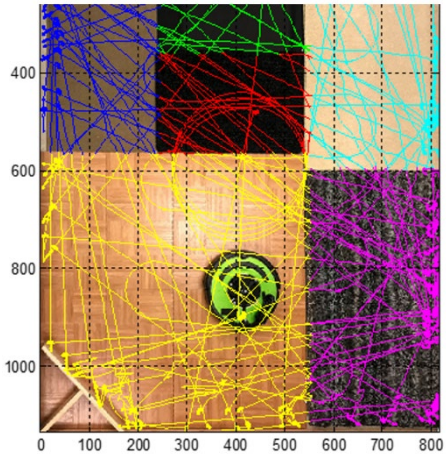

Figure 3-1 Robot motion capture

3.2 Life Imitates… the Test Environment

To ensure our robot's success, it should be tested under conditions similar to its intended deployment environment - whether that's a mock warehouse, mock hospital, or mock living room. Environments vary in where objects are placed, the length of hallways, changes in flooring, magnetic fields, and temperature. Ensuring that your test environment can cover these types of changes builds a more confident and robust solution.

For example, CEVA's robot vacuum testing is done in a mock living room based on an international standard. This very specific standard contains multiple pieces of furniture, changes in flooring, inclines, bumps, and even requirements for what is on the walls (this is relevant for VSLAM robots). By using this set of obstacles and settings, we can collect heading and trajectory data for the same scenarios that one would see during use.

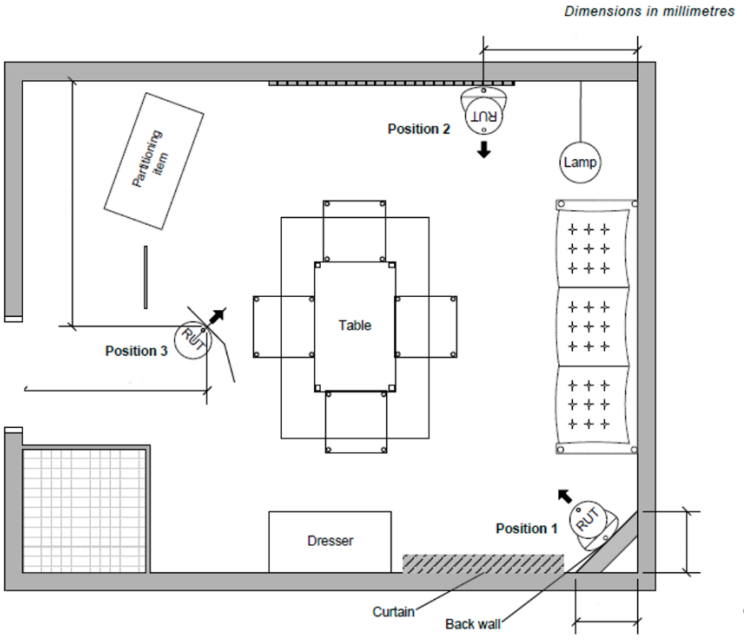

Figure 3-2 Robot vacuum test room

3.3 Expecting the Unexpected

As we mentioned in Section 2, inertial sensors are affected by temperature changes and these effects lead to heading error. The end use case environment will vary within a certain temperature range. Additionally, when our automated robots go out into the real world, they will literally run into issues outside of our test environment. Humans or animals will bump into it and the paths it takes might be filled with more obstacles than we model.

Testing needs to reflect these scenarios to be robust. The more iterations, the more complete our picture, and the better we can tweak our algorithms.

An example test plan might include the following tests:

- Baseline – changing temperatures within expected range inside the test environment
- Bump – adding sudden orientation change or displacement to emulate incidental bumps
- Obstacles – increasing the number of objects and disruptions
- Longevity – increasing testing run-time to emulate use in an industrial setting

Ideally, you want to run each of these tests multiple times on multiple testing platforms to gather as much data as possible and increase confidence in your work. This is obviously more costly and complex if you are testing the whole robot. With our focus on sensors, CEVA tests multiple sensors riding on the same robot to get as much data as we can. This allows us to track heading relative to truth with multiple sensors and glean more insight into how the base and outside factors affect their performance. If you're seeing a pattern here, it's because we like our data. The same principles we use for our sensor characterization are present in our testing as well.

4. Analysis and Conclusions

4.1 Reading the Lines

Data is nothing without the proper analysis, but with careful curation we can optimize our robot's tracking performance. For instance, with our comprehensive test plan, we can look at how fast the robot's perceived heading is drifting away from the heading measured by our motion capture system. You can miss valuable insights if you only summarize heading accuracy with a few numbers (like the heading difference at the end of each trial), because sometimes there can be large errors that are later cancelled out by other large errors, or maybe one result was good except for a brief glitch.

So instead, we examine the error growth rate at each moment in time (e.g., over a rolling 15s window) and treat each of these as a separate data point. Then we plot the distribution of these error growth values for each trial in a CDF (Cumulative Distribution Function) as seen in the example below. Looking at the plot (lines to the left are better here), we can easily compare the median performance vs the worst case or other percentile and identify outliers.

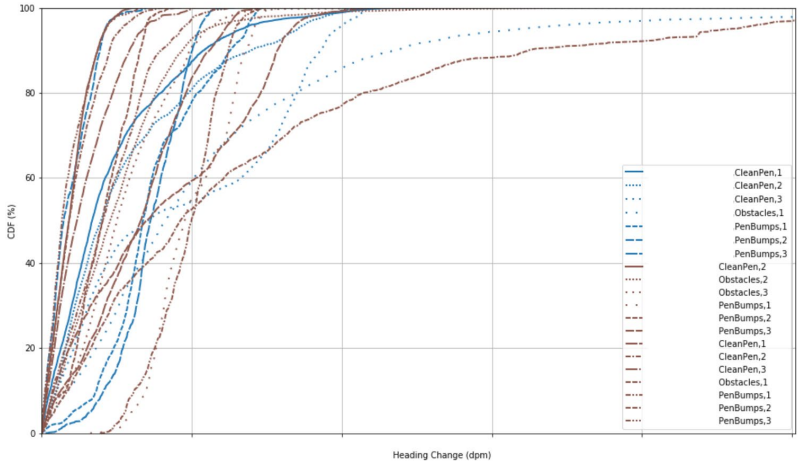

Figure 4-1 CDF of multiple algorithms over different conditions

This helps us determine which sensors and algorithms run with less heading drift than others and show us how to tweak the values for higher accuracy.

We perform similar analysis while looking at trajectory error. It can be measured in a few ways:

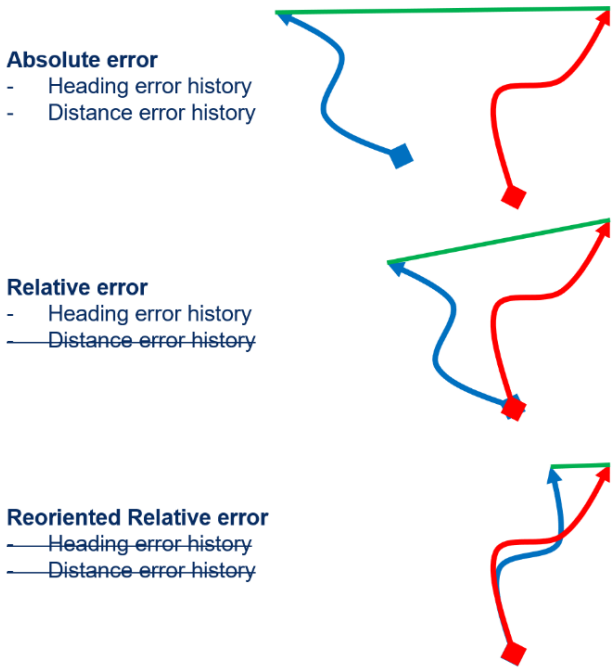

Figure 4-2 Trajectory error metrics

- Absolute error compares the end points of the perceived and actual trajectories independently. This allows you to see the accumulation of heading and distance error over the course of a long trial.
- Relative error adjusts the two datasets to the same starting point over each measurement window. This isolates the accumulation of previous errors from the error growth arising from heading error.
- Reoriented relative error accounts for translation and rotation differences at the start of each measurement window. This isolates the overall error growth per unit distance from previously accumulated errors. This is the most useful metric for identifying the source of trajectory errors, which appear as “hot spots” in the reoriented relative error.

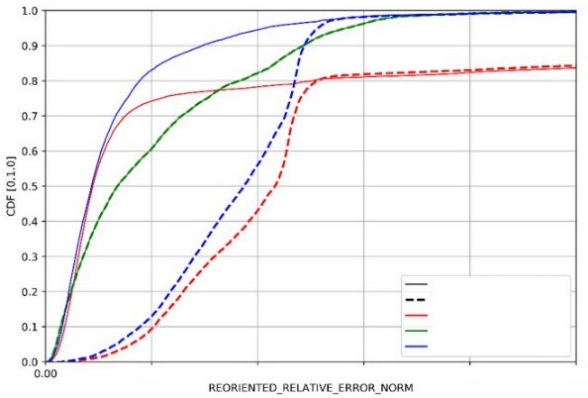

Figure 4-3 Relative error distribution for a variety of fusion methods and sensor conditions

4.2 Conclusion

The more complex the machine, the more understanding needed for each part the whole. Ensuring that you get the best sensors for your robot, making the right test plan, collecting data, and finding insight in the analysis is an intimidating endeavor. Hopefully, this paper has shown the complexity and impact of having the right tools in your arsenal.

However, if this process sounds overly complex for dealing with the sensors of your robot, you're in luck. CEVA has spent nearly 20 years studying and characterizing sensors to produce feature-packed sensor fusion solutions. Our dynamic calibration algorithms deal with ZRO and ZGO over varying temperatures in real time so you don't have to think about them. Gyroscope scale is easily corrected with our per-device calibration algorithms. And our interactive calibration algorithm minimizes sensor drift to significantly reduce drift in low cost sensors in robotic applications.

As of writing, we've released a new robotic dead reckoning product called MotionEngine™ Scout that intelligently fuses the data between wheel encoders, an optical flow sensor, and IMU. This fusion from Scout also cross-calibrates each sensor using information from the others. Scout achieves trajectory accuracies 5-10x better than optical flow sensor or wheels alone. Our sensor fusion simplifies the complexities of working with various sensors, and instead provides OEMs a simple interface for their robot navigation.

With our thorough testing, comprehensive data collection, and advanced analytics, CEVA has the sensor expertise to exceed your sensor needs. If you're interested in learning more about our robotic offerings, services, or any other application specific features that CEVA provides, please contact us.

5. Glossary

Table 3-1 defines the acronyms used in this document.

Table 3-1: Acronyms

Term	Definition
CDF	Cumulative Distribution Function
IMU	Inertial Measurement Unit
SLAM	Simultaneous Location and Mapping
VSLAM	Visual SLAM
ZGO	Zero-G Offset (Accelerometer bias)
ZRO	Zero-Rate Offset (Gyro bias)