Moving Baseline Localization for Multi-Vehicle Maritime Operations

Jonatan Scharff Willners*†, Pedro Patron* and Yvan R. Pettilot†

*SeeByte
30 Queensferry Road, Edinburgh EH4 2HS, UK
Email: {jonatan.willners, pedro.patron}@seebyte.com

†School of Electrical Engineering
Heriot-Watt University, Edinburgh EH14 4AS, UK
Email: y.r.pettilot@hw.ac.uk

Abstract—**Development in autonomous systems in recent years have enabled an increase on multi-vehicle maritime operations. Air, surface and underwater vehicles are now able to cooperate to jointly accomplish the objectives of a shared mission plan. In multi-vehicle scenarios, knowing accurately the platforms position is of great importance. If the navigation error is not controlled, unexpected and undesirable events such as collisions, less reliable data or loss of platforms can occur. In environments where Global Positioning System (GPS) is denied, such as underwater, updating the global position for the platform is difficult and often requires taking specific actions which are not part of the original mission. In the underwater domain, this typically means getting a GPS fix on the surface. Breaching the surface is a time consuming and potentially dangerous or unfeasible task. In this paper a framework striving to reduce or even completely remove the need for an Autonomous Underwater Vehicle (AUV) to surface for GPS fix is described. The framework proposed is decentralized and opportunistic. It is based on a moving long-baseline with One-Way-Travel-Time (OWTT) for range measurements and provides the capability to synchronize clocks between different platforms.**

Keywords*—*Localization, moving long-baseline, clock synchronization, multi-vehicle, marine robotics.

I. Introduction

Multi-vehicle missions can range from mine countermeasure operations performed by the Navy to the tracking of subsea features in Oceanographic expeditions. These operations require platforms to be deployed for long periods of time. Missions can span from hours to weeks. Vehicles have to be able to perform unattended without the support of a vessel continuously injecting navigation updates via acoustic messages. Under these circumstances, navigation solution of the underwater vehicles drifts over time. This increases the uncertainty of vehicle's location and, as a consequence, the risk of getting lost. When this occurs, the current general approach is for the underwater vehicle to come to the surface to relocate itself using a GPS antenna. This procedure is disruptive, time consuming and risky. It requires the vehicle to pause its task in the mission. It requires the vehicle to spend valuable mission time transiting over the vertical plane instead of gathering data. Ultimately, it exposes the vehicle to the surface of the water, highly increasing its vulnerability.

A high precision in localization for autonomous systems is an essential requirement to perform safe multi-vehicle autonomous operations. In some environments, such as underwater, no absolute positioning system similar to GPS exists. This induces one more complexity to the system. In the underwater environment communication is limited to low bandwidth, no radio-signals can be used, therefore no access to GPS to get an external update of the platforms position. Instead the platforms rely on dead reckoning. This contributes to a growing error. From Eustice *et al.*'s paper [1] it can be seen that a 48 hour dead reckoning with 1% error at a speed of 3 knot would result in an error in position of 2.6km. Such errors could mean that obtained data is useless or in worse case a loss of the platform. Another way to get an update on position, when GPS is unavailable, is to have a remote source provide information to the platform to help it regain certainty in its position. The most common way for this is to use an ultra-short-baseline (USBL) or long-baseline (LBL). USBL have limited use in multi-vehicle scenarios as the update rate for each vehicle decreases as 1/N for a N vehicle scenario.
When LBL is used, it can be by either fixed or moving transponders. Fixed transponder limits the area to 5-10km from the transponders, which can be too small in many cases. Moving Long Baseline (MLBL) systems have been used successfully in different trials such as in [7], Vaganay et. al. used two surface vehicles to aid an AUV with the help of ranging. In [1] [2], Eustice *et al.* uses time-synchronized platforms to perform One-Way-Travel-Time (OWTT) once every 5-20 seconds. In the paper the authors uses one ASV to help one AUV to keep its confidence in its position. This paper focuses on how to use a similar approach to more vehicles simultaneous and with sparser time between sending messages. In [3], Fallon *et al.* uses three autonomous surface vehicles (ASV) to update each other as if they were underwater, they use the GPS to keep a synchronized clock between the different vehicles. Simulations with 16 AUVs using the same cooperative localization framework have also been done. Updating is done by giving each vehicle a time-slot to broadcast its data to help the other platforms to update their position. Synchronized clocks are essential to perform OWTT. Kebkal et. al. [6] performed clock synchronization with 4 modems, with stationary positions up to 1634 meters from each other, as well as measuring OWTT of signals sent

between them over a 24 hour period. In their paper different time synchronization methods are discussed, such as precision time protocol, reference broadcast synchronization[9], timing-sync protocol for sensor networks[10]. Most of the time synchronization protocols are not directly applicable to the underwater domain. They do not take in to account the signal propagation time. This time can be negligible for radio signals but acoustic signals travel at a five order magnitude slower. Different methods to filter a position based on range only measurement was compared in [8] where R. Diamant and L. Lampe came to the conclusion that of the compared methods, (Extended Kalman Filter(EKF), Non-linear Least Squares (NLS) and Particle Filter(PF)). NLS and PF showed superior results than EKF. Their test using NLS showed a reduction of about 80% in the error of the vehicles position compared to the vehicles own navigation. Another alternative approach to get ranging data is Two-Way-Travel-Time(TWTT) where one platforms sends out a request marking the time the message was sent, then waits for a response and saves the time. This would give the time the signal traveled, and half of that feed in to a model of speed propagation in water would give the distance between vehicles. This is a fairly easy and robust method, it does not rely on the clocks being synchronized between vehicles. The drawback is the same as for USBL, it does not scale well with multi-vehicle scenarios. As this paper is focused towards multi-vehicle OWTT will be the main source for ranging data. As one platform can feed all other in range of the modem with one single broadcasted message. TWTT will be used in some cases as a comparison OWTT ranging measurements.

The rest of the paper is organized as follows, Section II describes the frameworks different parts as well as the experiments and simulations performed. III concludes the paper and IV show the future work of the described framework.

II. System description

The goal for this paper is to present an increased localization for platforms in multi-vehicle systems. The proposed framework uses trilateration in an opportunistic and decentralized manner. By opportunistic it refers to that the system does not expect any messages to arrive nor messages to arrive at a certain period but instead if a messages containing the correct type of data is received it will use the data included in the message. By decentralized it is meant to show that the system does not rely on any form of dedicated master, but instead any vehicle can feed updates to the network to be used for trilateration. The system is supporting multi-vehicle scenarios in the way that vehicles can send messages used for trilateration between each other, keeping a local localization between the vehicles known. At any time one vehicle can surface to update the systems localization with an absolute reference. The absolute position of the system can also be done with platforms not currently in the system such as an ASV moving to the platforms expected location and performing a pattern to feed the system an accurate position, letting the vehicles continue their objectives without being interrupted to surfacing for GPS fix. This can also be done with for example an Unmanned Aerial Vehicle(UAV) landing and feeding the system before taking of. This decentralized approach makes the system versatile and robust towards changes in the system configuration, such as the loss of a platform. An advantage of making the system this way is that platforms can be added and withdrawn from the system without affecting the remaining platforms, since no platform is relying on a specific other for the trilateration. To be able to perform a reliable trilateration to get an estimated position, accurate range measurements between platforms are required. The method to perform ranging in this paper is to use OWTT, as it can handle an arbitrary amount of vehicles compared to TWTT. For OWTT to work the different platforms needs to be synchronized in time to be able to calculate the range depending on the time difference from when the signal was sent and when it was received. With a signal's travel time, it can be converted to a range, by using a model speed of sound in water. The relative movements of the different platforms can be an factor that contributes to an error in the solution. The worst cases is if movement between the platforms is parallel which gives more than one solution or if all platforms are static(with one transmitter) which makes the system unobservable due to giving infinity solutions on a circle around the sender. These problem can be partly solved by filtering result of the solution. For instance, in [11] it can be seen how taking the trajectory of a support vessel into consideration can help another platforms localization by increasing observability.

```
struct {
    float x, y, r;
} landmark;
```

Fig. 1: Landmark observations x and y position and calculated range r based on OWTT(calculated when message is received).

First, in section II-A an overview of how trilateration is used in the system is described and in II-B, the description and results of simulations is presented. Then in section II-C, the tested clock synchronization with data is showed.

A. Trilateration

Trilateration is a method of estimating the position of a platform based on measured ranges to other known landmarks. It can be used to find a position in N dimensions by using at least N+1 observations. In underwater scenarios the positions is in 3 dimensions, however, with most platforms equipped with a pressure sensor, an accurate estimation of depth is already known making the problem 2 dimensional instead. Trilateration problems can be solved with different approaches, some of which have been compared in [4],[5] and [8], demonstrating that Non-linear Least Squares(NLS) is the better solver, when compared to Particle Filter and Extended Kalman Filter. For that reason NLS was chosen as the method to solve the trilateration problem in this paper. The framework is event based and computes the trilateration when a message containing the correct data is received. The pseudocode for how an incoming message is handled can be seen in section II-A1. Simplified data structures, *landmark* and *observationData*, used to explain the algorithms can be seen in figures 1 and 2.

1) New incoming message: When a new message is received it is checked if it contains values for position in x and y as well as a time stamp. If it does, it can be used for trilateration and the message's data will be added with

```
struct {
    landmark t, r;
} observationData;
```

Fig. 2: Transmitter data t and receiver data r. Created when a message applicable for trilateration is received.

algorithm 2. If enough observations (minimum of 3) have been received the trilateration can be performed. To perform the trilateration the transmitter's position in the saved observations needs to be compensated for which can be seen in algorithm 3. With the compensated positions used as input in the algorithm to solve the NLS equation as seen in equation 1, which in its turn solves for a new position of the vehicle and updates the model of the vehicle with these values.

```
Input: incMsg
if $newMessage.contains(x, y, timeStamp)$ then
    $addNewData(incMsg)$;
    if $length(observedLandmarks) >= 3$ then
        $c =\leftarrow compensatedData()$;
        $new\_x, new\_y \leftarrow NLSSolver(c)$;
    end
end
```

Algorithm 1: Pseudocode for when a new message is received

The pseudocode for addNewData can be seen in section II-A2, compensatedData in II-A3 and NLSSolver in section II-A4.

2) addNewData: When new data is received that contains the transmitters position in x, y and the time stamp for when the message was sent, a new observation to the list of *observationData* can be added. The new data contains the transmitters position and the distance calculated for the message based on the OWTT gained from the time stamp. The other part of the new data is the receivers position, which is later used to compensate the positions of the transmitters position when performing trilateration.

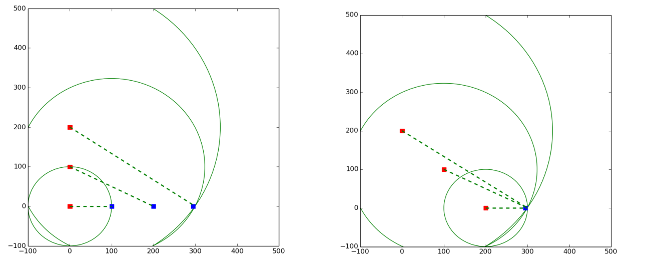
```
Input: incMsg
$transmitter \leftarrow landmark()$;
$transmitter.r \leftarrow$
  $RangeFromTime(incMsg.timeStamp)$;
$newObservation \leftarrow obeservationData()$;
$newObservation.t \leftarrow transmitter$;
$newObservation.r \leftarrow getSelfPos()$;
$addNewObservationToList(newObservation)$;
```

Algorithm 2: addNewData(). Adds new data to the observedLandmarks list.

3) Transmitter position compensation: Before the data can be solved for a new position by trilateration with NLS the transmitter positions need so be compensated. This is to model the signals as if all was received at the current time. An illustration of this can be seen in figure 3. Figure 3b show how the transmitter positions have been moved with the relative vector of the receiver vehicles positions in between received messages. The algorithm for this can be seen in algorithm 3.

```
Input: A list $a$ of observationData of length n
Output: The compensated list $c$ of landmark
for $i \leftarrow 0$ to $n$ do
    $c[i].t.x \leftarrow a[i].t.x + a[i].r.x - a[n].r.x$;
    $c[i].t.y \leftarrow a[i].t.y + a[i].r.y - a[n].r.y$;
    $c[i].t.r \leftarrow a[i].t.r$;
end
return $c$;
```

Algorithm 3: compensatedData(). The algorithm compensating transmitter positions by receivers relative motion to current time.



(a) The original transmitter positions (b) The compensated transmitter positions

Fig. 3: How transmitter positions is moved relative to receivers relative motion between observations to prepare to solve the trilateration problem with NLS. The red squares are the transmitters position when sending the message, the receivers position when receiving the message is the blue squares, the message's traveling length until received is the green circle/line.

4) Non-linear Least Squares: The NLS solver used in this paper is based on LmFit [16]. The algorithm takes a list of *landmark*s as input and returns the solution where *x* and *y* minimizes the sum of errors. The problem can be seen as in figure 3b, to find where the circles overlap. In the pictures there is no noise, so a real model would not necessary be as easy to solve, otherwise it would be easy to calculate where n circles with perfect measurements overlaps.

$$[h] \sum_{i=1}^{len(data)} (data[i].x - x)^2 + (data[i].y - y)^2 - data[i].r^2 \quad (1)$$

Where *data[i].x* and *data[i].y* is UTM coordinates of the transmitter and *data[i].r* is calculated signal's distance in meters, *len(data)* is number of observations used (minimum of 3 for finding a solution in 2 dimensions), the result is the calculated position of the receiver in UTM coordinates.

B. Trilateration simulation

A simulator to test the framework with different noise and drift to the system, both in platforms and signals was used. An image of the simulator can be seen in figure 4. The simulator works in two dimensions, as depth is considered a known value from sensors. The simulations for this paper uses five platforms, of which one is surfacing to gain an absolute position from GPS. The surfacing platform will then send OWTT messages to feed the system, in these simulations one message will be sent every 150 seconds. The four other platforms will continue to perform their objectives, in our scenarios a predefined lawnmower pattern. In this paper, simulations with two different drift and noise will be shown, both with higher error in dead reckoning than what is to be expected in a real scenario to determine the usefulness in more extreme scenarios. In figure 5 the platforms have a continuous drift in x and y at 2.5% and an added noise with of 2.5% of travel distance. As expected, the error for each platform grows until enough observations from the surfacing platform is obtained (at circa time 450) where the error goes down and becomes more stable in long term with an infused error due to dead reckoning between messages from the platform with high confidence. In figure 7 the platforms have no continuous drift in x and y but only a noise of 2.5% of travel distance. It is here shown how the result varies depending on how many observations that is used to solve the trilateration problem. In these the same random error is introduced to fairly see the difference. In the top row the last 3 observations is used, in the middle up to the last 4 and in the bottom up to the last 5. In all of the simulations, the error is reduced by the use of one platform sacrificing time to support the others, this platform could also be considered to be an arbitrary platform coming to the area to help feed the other platform.

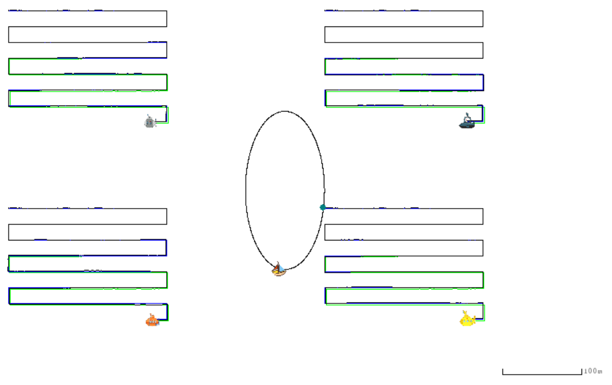


Fig. 4: Simulation environment with 5 platforms.

C. Clock-synchronization

Clock-synchronization in this paper refers to finding the offset between the internal clocks in the modems on different platforms. For synchronization Network Time Protocol(NTP) is used. The internal system clock of the used modems can not be set, so instead the offset between different platforms is saved and compensated for in the software when an action relying on time synchronization is performed. Since different clocks will not have the same clock skew, the platform needs to resynchronize its clocks with other platforms in the system on a regular basis. When a resynchronization is done, the clock skew between the two platforms performing NTP can be approximated. This clock skew can then be applied to calculate a more precise offset between clocks over time. The experiments were performed with two platforms, a transmitter responsible for sending out time stamped signals, and a receiver which handles incoming messages and performs NTP synchronizations.

1) (: Experiment 1: clock skew) The first experiment performed was to determine how the clocks on different platforms differ from each other over time, and not compensating for the clock skew.

The transmitter platform sends 49 messages to measure time difference by OWTT followed by 1 message to force the platforms to a NTP synchronization. The period in between each message is 5 seconds.

The receiver platform listens and performs NTP synchronizations and logs the incoming messages. The result can be seen in figure 6 as the cyan colored graph. As expected the error grows more or less at a constant rate until a new NTP synchronization is successfully performed which resets the error. It can be seen that the drift was reset at different periods and scale over the duration of the experiment. This was caused by failing to receive some of the NTP synchronization messages and clock drifts not being constant.

2) (: Experiment 2: clock skew compensation) With experiment II-C1 showing that the clock drift between the platforms are close to constantly increasing over time, the clock skew can be modeled and compensated for. The Master platform runs the same program as in experiment II-C1. But the Slave platform models the clock skew as seen in equation 2.

$$clock\_skew = \frac{(new\_offset\_NTP - last\_offset\_NTP)}{(time\_now - time\_last\_NTP)} \tag{2}$$

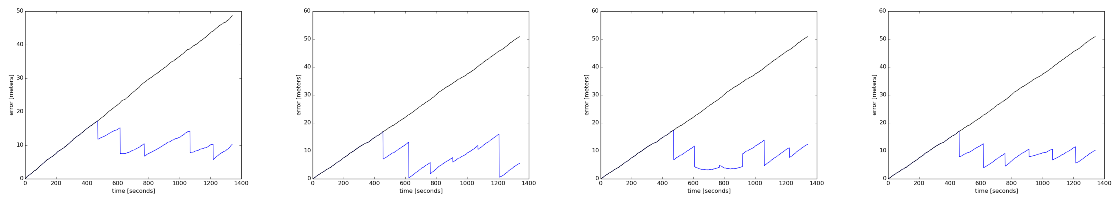
$$compensated\_time(t) = t + (t - time\_last\_NTP) * clock\_skew \tag{3}$$

The result of this can be seen in figure 6 as the black graph. As seen, the clock skew of the compensated time between the two platforms decreases the error over time. Other ways to do this would be to use linear regression over a set of NTP synchronizations, this have been done by Heideman *et al.* in [12], but the approach in that paper is using 25 exchanges of messages to model the clock skew, which in a multi-vehicle scenario could result is not realistic.

With the clock-skew calculated, equation 3 can be used to calculate a compensated time.

III. Conclusion

We proposed an opportunistic and decentralized trilateration system based OWTT. This paper focuses around experiments with one platform with an absolute reference acting as a master to feed the navigation solution of the other platforms. The system models the clock skew to perform clock synchronization and decrease the drift of clocks in between platforms. Simulations with high levels of drift and noise were used to demonstrate that the framework is able to keep the error lower than dead reckoning. Clock-skew modeling enabled reliable ranging between platforms that could lead to reduction in the frequency of clock-synchronization between platforms.



(a) AUV1 (b) AUV2 (c) AUV3 (d) AUV4

N=3

Fig. 5: Simulation with 2.5% drift in x and y and an added noise of 2.5% of distance traveled. This simulations uses the 3 last observations to solve the trilateration problem. Blue line is error in position with calculated trilateration. Black line is error in position by dead reckoning. The vertical axis shows absolute error in meters and the horizontal show time in seconds.

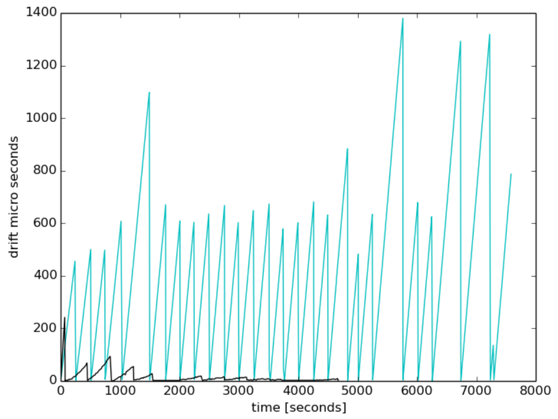


Fig. 6: Comparison between compensated(black) and uncompensated(cyan) clocks

IV. Future Work

Our future work will focus on exploring ways of extending the framework to a decentralized approach. This will require the use of for example weighted NLS and information filters for the transmitter to indicate how reliable the positional data transmitted is believed to be. This would be helpful when in a multi-vehicle scenario with mixed platforms with low and high accuracy in navigation is in the system. In such case the high accuracy platforms could improve to low without the need to surface to do so, and increase the knowledge of the local localization between platforms. As NTP does not take in to account changes of platforms position during message exchanges a more suitable clock-synchronization protocol will be used. Trials to show the framework used in real scenarios are to be performed.

References

[1] Ryan M. Eustice, Louis L. Whitcomb, Hanumant Singh, Matthew Grund, *Recent Advances in Synchronous-Clock One-Way-Travel-Time Acoustic Navigation*, Oceans 2006

[2] Ryan M. Eustice, Louis L. Whitcomb, Hanumant Singh, *Synchronous-Clock One-Way-Travel-Time Acoustic Navigation for Underwater Vehicles*, Journal of Field Robotics, Volume 28, Issue 1, Pages 121-136, 2011

[3] Fallon, Maurice F, Georgios Papadopoulos, and John J Leonard *A Measurement Distribution Framework for Cooperative Navigation Using Multiple AUVs* IEEE International Conference on Robotics and Automation, 2010. Pp. 4256-4263.

[4] William S. Murphy, Jr. , Willy Hereman, *Determination Of A Position In Three Dimensions Using Trilateration And Approximate Distances* 1999.

[5] Maurice F. Fallon, Georgios Papadopoulos, John J. Leonard and Nicholas M. Patrikalakis, *Cooperative AUV Navigation Using a Single Maneuvering Surface Craft* The International Journal of Robotics Research 29.12 (2010): 14611474.

[6] Konstantin G. Kebkal, Veronika K. Kebkal, Oleksiy G. Kebkal, and Roberto Petroccia, *Underwater Acoustic Modems (S2CR Series) for Synchronization ofUnderwater Acoustic Network Clocks During Payload Data Exchange* IEEE JOURNAL OF OCEANIC ENGINEERING, VOL. 41, NO. 2, APRIL 2016: 428-439

[7] Jrme Vaganay, John J. Leonard, Joseph A. Curcio, J. Scott Willcox, *Experimental Validation of the Moving Long Base-Line Navigation Concept* 2004 IEEE/OES Autonomous Underwater Vehicles (IEEE Cat. No.04CH37578): 59-65.

[8] Roee Diamant, Lampe Lutz, *Underwater localization with time-synchronization and propagation speed uncertainties* 2013, IEEE Transactions on Mobile Computing, volume 12, issue 7: 1257-1269

[9] J. Elson and D. Estrin, *Time Synchronization for Wireless Sensor Networks,* inProc. IEEEComput. Soc. 15th Int.ParallelDistrib.Process. Symp.,Washington, DC, USA, 2001, p. 186, ISSN: 0-7695-0990-8.

[10] S. Ganeriwal, R. Kumar, andM. B. Srivastava, *Timing-sync Protocol for Sensor Networks* in Proc. 1st Int. Conf. Embedded Netw. Sensor Syst., pp. 138149, DOI: 10.1145/958491.958508.

[11] Filip Mandić *Underwater navigation and localization using single range measurements* in Proc. 1st Int. Conf. Embedded Netw. Sensor Syst., pp. 138149, DOI: 10.1145/958491.958508.

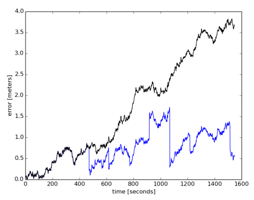
[12] Syed, Affan A. Heidemann, John *Time synchronization for high latency acoustic networks* April 2006

[13] https://www.evologics.de/en/products/sonobot/index.html [Accessed 2017-02-08]

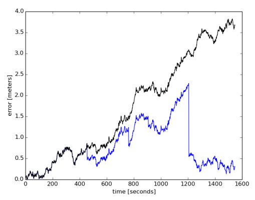
[14] https://www.evologics.de/en/products/acoustics/index.html [Accessed 2017-02-08]

[15] https://www.ros.org [Accessed 2017-03-15]

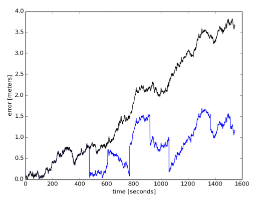
[16] https://lmfit.github.io/lmfit-py/ [Accessed 2017-04-19]



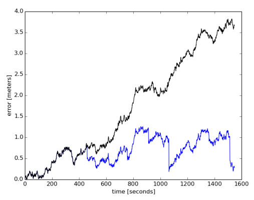
(a) AUV1



(b) AUV2

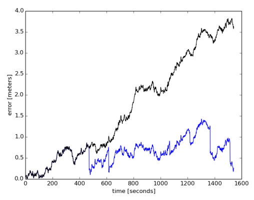


(c) AUV3

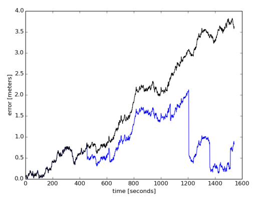


(d) AUV4

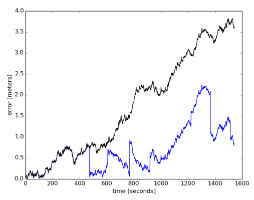
N=3



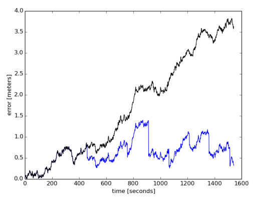
(e) AUV1



(f) AUV2

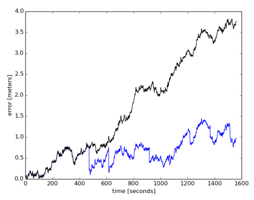


(g) AUV3

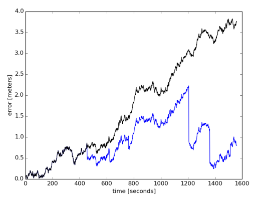


(h) AUV4

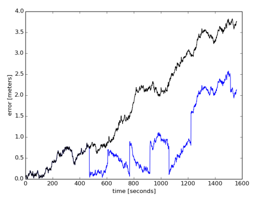
N=4



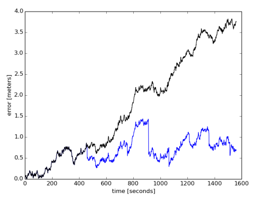
(i) AUV1



(j) AUV2



(k) AUV3



(l) AUV4

N=5

Fig. 7: Simulations with no drift in x and y and an added noise of 2.5% of distance traveled. Using the up to N most recent observations to perform trilateration. Blue line is error in position with calculated trilateration. Black line is error in position by dead reckoning. The vertical axis shows absolute error in meters and the horizontal show time in seconds.