VANDERBILT UNIVERSITY

INSTITUTE FOR SOFTWARE INTEGRATED SYSTEMS

Institute for Software Integrated Systems
Vanderbilt University
Nashville Tennessee 37235

TECHNICAL REPORT

TR #: ISIS-05-601
Title: Wireless sensor network-based projectile trajectory estimation
Authors: Gyorgy Balogh, Akos Ledeczi, Miklos Maroti

Wireless Sensor Network-Based Projectile Trajectory Estimation

György Balogh, Ákos Lédeczi and Miklós Maróti
Institute for Software Integrated Systems, Vanderbilt University
{gyorgy.balogh , akos.ledeczi , miklos.maroti}@vanderbilt.edu

Abstract

The report presents a sensor fusion algorithm that was utilized in a wireless sensor network-based acoustic countersniper system. The approach for fusing shockwave TOA data that are generated by supersonic projectiles is able to reconstruct the trajectory with an accuracy of 1 degree in both azimuth and elevation for long range shots. If a few muzzle blast detections are also available then accurate range estimation is also performed. The system performance was demonstrated multiple times at different MOUT (Military Operations in Urban Terrain) facilities of the US Army.

1 Introduction

The firing of a typical rifle results in two acoustic phenomena. The muzzle blast produces a spherical wave front, traveling at the speed of sound from the muzzle of the gun. The shock wave is generated in every point of the trajectory of the supersonic projectile producing a cone-shaped wave front (assuming the speed of the projectile is constant). The sensors of our countersniper system autonomously detect the shockwave and/or the muzzle blast, measure their times of arrival (TOA), and send the measured results to a base station through the ad-hoc wireless sensor network. Then the fusion algorithms running on the base station determine the location of the shooter and the trajectory of the projectile.

The sensor nodes are based on the UC Berkeley MICA2 mote device running the TinyOS embedded operating system [2], a widely used component-based architecture targeting scalable wireless sensor network applications. Each MICA2 mote is furnished with an ATmega 128L 8-bit microcontroller with 128 Kbytes instruction memory, 4 Kbytes data memory and typical embedded peripherals built in. The on-board radio transceiver operates in the 433 MHz ISM band and has a maximum transfer rate of 38.4 Kbits/sec with the maximum range of about 300 feet.

Real-time detection and classification of acoustic events require computational resources not available in the sensor node. Thus, we have designed an application specific sensor board built around a low-power fixed point ADSP-218x digital signal processor running at 50Mhz. Its internal program (48KB) and data (56KB) memory buffers with advanced addressing modes and DMA controllers enable sophisticated acoustic signal processing and advanced power management methods.

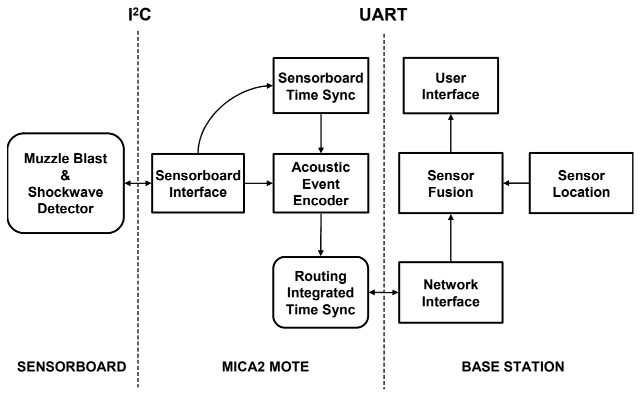

Figure 1. Software Architecture

Two independent analog input channels with low-cost electret microphones pick up the incoming acoustic signals utilizing a 2-stage amplification with software programmable gain (0-54dB). The A/D converters sample at up to 100kSPS at 12-bit resolution. Analog comparators with software adjustable thresholds can be used to wake up the signal processor from low-power sleep mode, enabling continuous operation for weeks on two AA batteries. A standard I2C bus connection and programmable interrupt and acknowledgement lines enable integration with the MICA2 mote [1, 3].

The software architecture of the system is depicted in Figure 1. The Muzzle Blast and Shockwave Detector is running on the DSP of the custom sensor board board. The TOA data from the board is sent through the I2C interface to the mote. The Acoustic Event Encoder assembles a packet containing the TOA data and passes it to the Routing service.

In addition to transporting the packets to the base station through multiple hops, the Message Routing service also performs implicit time synchronization as described in the next section. The Base Station runs the Sensor Fusion algorithm utilizing known sensor positions and displays the results on the User Interface.

1.1 The fusion of time

The countersniper system does not use a continuous time synchronization service providing a virtual global clock. From an application standpoint it is desirable that the system operates in a stealthy manner; it should not use any radio operation when not necessary. This also saves power, a precious resource in wireless sensor networks. But how can we correlate multiple observations in the absence of a global clock?

Our implicit time synchronization approach is based on the Elapsed Time of Arrival (ETA) algorithm [4]. The ETA algorithm works by storing the time of all events using the local clock of the node. Local-time conversions occur whenever messages are transmitted from one node to another. This is enabled by our highly accurate time stamping service that is part of the MAC layer of the radio stack of the mote [5]. At transmission time the elapsed time since the event occurrence is computed and included as a separate field in the message. On receipt, a node computes the event time by subtracting the elapsed time from the receiving node's local time.

The Routing Integrated Time Synchronization (RITS) protocol, is an extension of ETA over multiple hops by integrating it into the Directed Flood Routing Framework (DFRF) [6]. After detecting an event, the sensor node provides RITS with the local time of the event occurrence and a data packet. In the countersniper system this is a single bit differentiating between muzzle blast and shockwave. RITS, configured for convergecast, sends the packet to the base station along a multi-hop path while maintaining the event time using ETA. At each hop it converts the event time expressed in the local time of the sender to that of the receiver. When the message arrives at the base station the detection time of the acoustic event is provided to the sensor fusion algorithm using the *base station clock* in a transparent manner. Different observations of the same event then can be accurately correlated.

We observed that the maximum pairwise error in a 10-hop 60-node setup of MICA2 motes was well under 100 microseconds. As the sound travels approximately 3 cm-s in this time, it has a negligible effect on the overall localization accuracy.

2 Shockwave fusion

The muzzle blast fusion algorithm works very well when the acoustic source is located within the sensor field and there are at least 8-10 line-of-sight measurements. Once the shooter is shooting outside of the field the accuracy starts to decrease. One reason is that the angle of the sensor field from the shooter ("field of view") is getting smaller and hence, individual measurement errors have larger effect on the result. The other reason is that as the distance to the shooter increases less and less sensors are able to detect the muzzle blast at all. Once the shooter is beyond 30-50 meters muzzle blast alone is not enough to make accurate localization with our system.

Notice that the shockwave does not have this problem since it is generated by the projectile and not the gun. The sensor fusion is presumably deployed in and around the protected area. As long as the bullet flies in this area, there will be plenty of detections. For these reasons we have developed a shockwave fusion algorithm that is used for long range shots. In practical terms, when the number of muzzle blast detections are not enough (i.e. < 8) for accurate localization, the shockwave fusion outlined below is applied. The two different fusion approaches could be run in parallel as well.

The shockwave caused by a supersonic projectile is the result of the air being greatly compressed at the tip of the bullet as it slices through the air. A broadening wave of the compressed air trails out diagonally from the tip and all the sides of the bullet as it moves forward, creating a conical waveform. See Figure 2. The shockwave front travels at the speed of sound while the bullet travels faster. The angle of the cone depends on the ratio of the speed of the bullet and the speed of sound:

$$\alpha = 2\arcsin\left(\frac{v_{sound}}{v_{bullet}}\right)$$

Note that this angle is constantly changing as the bullet decelerates and, in fact, the shock wave is not a cone at all. However, we assume constant bullet speed to simplify the necessary calculations. As it turns out, the overall accuracy of the system is still remarkable.

Sensors can identify the shockwave based on its unique waveform and the significant energy level. If enough sensors make detections and their positions and shockwave time of arrivals are known, the bullet trajectory can be computed. However, the same arguments about noisy and erroneous measurements and multiple simultaneous shots that were made in the previous section are equally valid here. Thus, we have applied a numerical approach for this fusion problem as well.

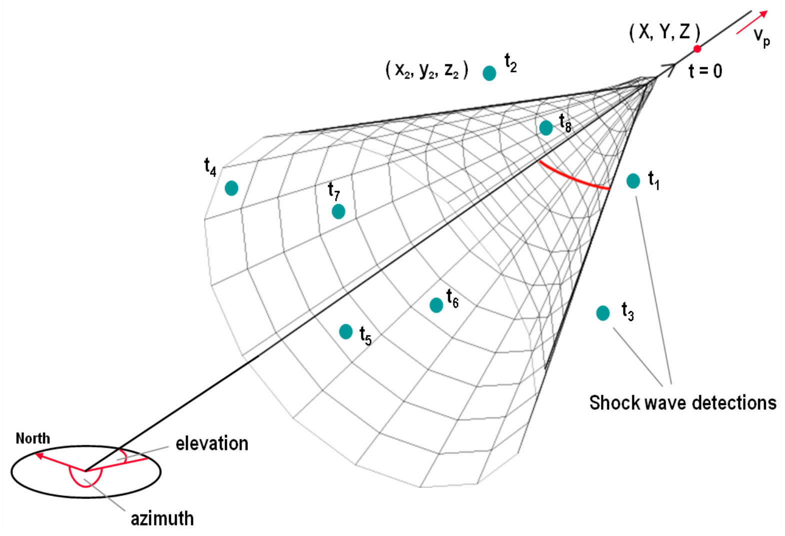

Figure 2. Shockwave front

The consistency function based solution can be generalized and applied here as well. In fact, the muzzle blast and shockwave measurements could be applied at the same time contributing to the same consistency function. However, the number of dimensions would increase significantly. In addition to the three coordinates of the shooter position (x, y, z) and the shot time t, we have the two angles for azimuth and elevation and the speed of the projectile. The search in this 7-dimesional space is currently computationally infeasible. Instead, we have applied a genetic algorithm for the task at hand.

A real-time fusion algorithm receives shockwave measurements from the sensor network. First the algorithm splits the received measurements into smaller groups representing single shots or possibly almost simultaneous multiple shots. Then a genetic algorithm searches for the trajectories in each group that match the detections the best. Once the best trajectories have been found, a range estimation algorithm computes the estimated ranges using muzzle blast measurements if available.

2.1 Slicing

The slicing algorithm is responsible for splitting the incoming measurements into relatively small, processable groups. The online algorithm has three major states: idle, collecting and computing. When a new measurement is received, the system goes into the collecting state and the collection of the measurements begins. It stops collecting and goes into the processing state after a fixed amount of time ($T_{\max\_collecting\_time}$) elapsed form the receiving of the first message, or a fixed amount of time ($T_{\max\_silence\_time}$) elapsed without receiving any more messages.

In the computing state, the first step is to split the shockwave measurements into smaller groups. Let D be the maximum of pairwise distances between sensors in the field. If there are two shockwave measurements which have a time difference larger than $D / v_{speed\_of\_sound}$, then the two detections cannot be the result of a single shockwave, because the wave front travels at the speed of sound and the projectile even faster.

The shockwave measurements are sorted by time and split into groups if and where there are larger time gaps than $D / v_{speed\_of\_sound}$.

2.2 Trajectory search

Given n shockwave measurements: $s_1(x_1, y_1, z_1, t_1), s_2(x_2, y_2, z_2, t_2), \dots, s_n(x_n, y_n, z_n, t_n)$ where x_i, y_i, z_i are the coordinates of a sensor and t_i is the corresponding TOA of the shockwave, we are looking for m trajectories: $tr_1(X_1, Y_1, Z_1, \alpha_1, \beta_1, v_1), \dots, tr_m(X_m, Y_m, Z_m, \alpha_m, \beta_m, v_m)$ which can generate the given measurements, where X_j, Y_j, Z_j is a point in space where the trajectory goes through, α_j is the azimuth, β_j is the elevation and v_j is the speed of the projectile. The X_j, Y_j, Z_j point of the trajectory is the point where the bullet was at time t_0 where $t_0 = \frac{1}{n}\sum_{i=1}^{n} t_i$. This is a somewhat arbitrary choice for a single point on the trajectory that is close to the sensor field.

For a given $tr_j(X_j, Y_j, Z_j, \alpha_j, \beta_j, v_j)$ trajectory and a $s_i(x_i, y_i, z_i, t_i)$ shockwave measurement the theoretical time of arrival of the shockwave at the sensor (t'_i) can be calculated using simple Euclidean geometry.

From the measured and the theoretical time of arrivals a simple least squares error function of the trajectory is calculated. A genetic algorithm (GA) has been used to find the trajectory with the smallest error. Here is the brief description of the GA applied:

1. Generate an initial population of *n* random trajectories
2. Select *m* individuals randomly from the population
3. Evaluate each individual in the selected subset using the error function (described later)
4. Sort the subset according to error
5. Remove the worst 20% of the individuals in the subset than generate new individuals by selecting random parents from the best 20% and applying genetic operators on the parents
6. Go to 2

Typical parameters used were n = 5000 and m = 500. A general problem with GA is that it can get stuck in a local minimum when the whole population becomes homogeneous containing the same exact individuals. Different heuristics have been proposed to avoid this problem. We use the tournament selection technique. We select a smaller subpopulation randomly at each step, this gives a chance for individuals with worse error value also to breed and makes the homogenization slower.

What happens if there are multiple shots or erroneous measurements (i.e. echoes)? In this case we cannot use all the measurements, we need to identify those that are good measurements and belong to the same shot. Therefore, we have extended the representation of a trajectory with a subset of the measurement indexes. This way the GA not only searches for the trajectory, but for a set of measurements as well, where the trajectory has the smallest error. Once the best trajectory is found, the corresponding events are removed from the group and the search is started again on the rest. It is repeated until the number of events becomes too small to be able to define a new solution. A trajectory has 6 parameters, therefore, at least 6 measurements are needed to be able to find the trajectory.

The representation of a possible solution is the following: $sol(x, y, z, \alpha, \beta, v, S)$ where $x, y, z, \alpha, \beta, v$ is the trajectory parameters and *S* is a subset of numbers from *1..n* (shockwave measurement indexes).

The following error function has been used to evaluate a given $sol(x, y, z, \alpha, \beta, v, S)$ solution:

$$error = \begin{cases} \dfrac{1}{\left| i \in S \vee |t_i - t'| < T \right|} \sqrt{\sum_{i \in S \vee |t_i - t'| < T} (t_i - t'_i)^2} & |S| >= K \\ \infty & |S| < K \end{cases}$$

If we have fewer than *K* measurements selected then the error is infinite. Otherwise, we sum the squared difference of all the measurements selected in the subset and the ones that has smaller difference than *T*. This second extension part is needed because otherwise the GA would just decrease the number of measurements in the set to *K* because leaving a measurement out always decreases the error. We set *K* to 7 after experimentation with field data. If we choose a small *T,* not all the measurements of the same shot will be selected. If we set it to too large, measurements from other shots can be included, increasing the error and distorting the result. If we know the localization error of the sensors and the typical shockwave detection error, we can estimate the largest measurement error in one shot and we have to choose that value for *T*.

2.3 Range estimation

Once we know the trajectory and we have a few muzzle blast measurements, we can estimate the range to the source. Again, special care must be taken because of potential multipath measurements and multiple shots.

The trajectory gives us not only a line in space but also a timeline of the bullet, because we know the trajectory, the speed of the projectile and a point in both space (X, Y, Z) and time (t) of the projectile's location. As the location of the muzzle blast is also on the trajectory, we just have to correlate the trajectory and the muzzle blast TOA data.

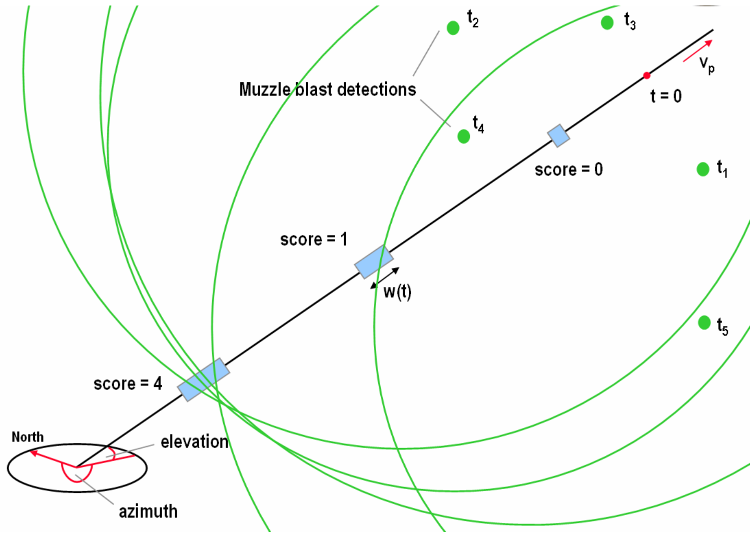

Figure 3. Range estimation

To compensate for erroneous detections and multiple sources, we just simply count the number of muzzle blast measurements that support a given point in space and time on the trajectory. We slide a small window backwards on the trajectory which gives us the possible space regions. The width of the window is determined by the estimated detection errors. A typical number is 1 meter. The time corresponding to the window location can be easily calculated form the speed of the bullet. A muzzle blast event is consistent with the current window if

$$d_{\min} < (t_m - t_w)v_{sound} < d_{\max}$$

where $d_{\min}$ ($d_{\max}$) is the minimum (maximum) distance of the sensor from the window on the trajectory, t_m is the TOA measurement and t_w is the time corresponding to the current window location.

The window with the maximum number of consistent muzzle blast detections gives us the estimated origin of the shot and, hence, the range.

2.4 Evaluation

To test the accuracy of the shockwave fusion algorithm, we have conducted field experiments in a US Army facility. The sensor network of 60 motes covered an 80x80m area. The motes were placed on surveyed positions with an estimated accuracy of 20 cm. About half of the motes were on the ground, the rest were placed at elevations up to 6m. Shooter positions were also surveyed. They were located up to 100m from the edge of the sensor field. Two locations were in the basket of a cherry picker at approximately 12m height. Various targets were placed inside and outside the sensor field opposite from the shooter positions. 12 shots have been fired over the middle of the network shooting approximately 100 meters from the edge of the sensor field, so there where sensors each side of the trajectory. The average azimuth error was 0.66 degree, the average elevation error was 0.61 degree, and the average range error was 2.56 meters. Another 11 shots were fired from the same distance over the edge of the network, so there were no or only a few sensors on one side of the trajectory, the average error increased to 1.41 degrees in azimuth, to 1.11 degrees in elevation and to 6.04 meters in range. Multiple simultaneous shots have also been tested with mixed results. This area needs further research.

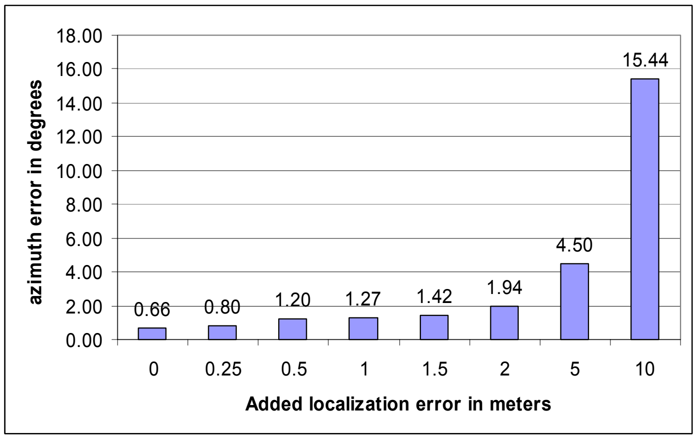

Figure 4. Sensitivity to sensor location error

The calculation of a single shot trajectory takes about 3-4 seconds on a 3GHz PC. To test the sensitivity of the fusion algorithm to sensor location error, we have conducted a preliminary experiment using the 12 shots fired in the middle of the network described above. We added additional errors to the sensor location data used during the field test. For each shot and for each coordinate (x, y, z) of each sensor position, a random error was added using a uniform distribution between 0 and d_{max}. The results for azimuth accuracy are summarized in Figure 4. For each maximum additional sensor location error, each shot was tested ten times using different sensor locations. Therefore, each bar in the figure represents 120 experiments. The apparent insensitivity of the fusion to sensor location error is encouraging for the future development of the system.

3 Acknowledgement

We would like to gratefully acknowledge the sponsorship of the DARPA/IXO NEST program.

4 References

[1] Simon, G., et al.: Sensor Network-based Countersniper System. *In Proc of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys),* November 2004.

[2] Hill, J., Culler, D. Mica: A Wireless Platform for Deeply Embedded Networks. IEEE Micro, Vol. 22, No. 6, 2002, pp. 12–24.

[3] Lupu, E. et al. Speaker Verification Rate Study Using the TESPAR Coding Method. In *Proc. of COST 276 Workshop on Information and Knowledge Management for Integrated Media Communication*, 2002

[4] Kusy, B. et al. Elapsed Time on Arrival: A simple, versatile, and scalable primitive for canonical time synchronization services, submitted to *Int. J. of Ad Hoc and Ubiquitous Computing*, 2005

[5] Maroti M., Kusy B., Simon G., Ledeczi A.: The Flooding Time Synchronization Protocol, in *Proc of ACM Second International Conference on Embedded Networked Sensor Systems (SenSys 04)*, pp. 39-49, Baltimore, MD, November 3, 2004.

[6] Maroti M.: Directed Flood-Routing Framework for Wireless Sensor Networks, *Middleware 2004*, pp, Toronto, Canada, October, 2004