ISIS Technical Report
ISIS-14-105

Hough-transform based acoustic multi-shooter localization and data association

12 December, 2014

Janos Sallai, Akos Ledeczi
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN, USA

Hough-transform based acoustic multi-shooter localization and data association

Janos Sallai and Akos Ledeczi
{janos.sallai,akos.ledeczi}@vanderbilt.edu

Institute for Software Integrated Systems
Vanderbilt University

Abstract

In this report, we consider the application of Hough Transform (HT) based methods to attack two problem areas in acoustic shooter location: i.) robustness against non-Gaussian measurement errors (e.g. echos) and ii.) the data association problem of multiple simultaneous shots.

In real-life scenarios, it is often the case that only a subset of the reporting sensors are in line-of-sight (LOS) situation, reporting correct detection results with some noise that can be assumed to be Gaussian. However, nodes in non-line-of-sight (NLOS) situation may detect erroneous signal arrival times and angles. Such detections often differ significantly from the expected (ground truth) values, and may even outnumber the correct detections, depending on the geometry of the shot and the sensor deployment.

Conventional methods, e.g. closed-form solutions, nonlinear optimization algorithms, statistical techniques often fail if the error pattern of the inputs is not Gaussian, or require extensive tuning to achieve acceptable results.

Such methods typically struggle with another common phenomenon: multiple simultaneous shots. When two or more shots are fired simultaneously such that the sensors are detecting multiple acoustic events within a short time window, the order of wavefronts arriving at different sensors may be different due to the geometry. That is, it is not possible to sort out which detection belongs to which shot. Before invoking the shooter location and trajectory solver, a data association problem must be solved that pairs the shots with the corresponding detections.

We attack the above two problem areas with Hough transform (HT) based techniques, a family of methods in image processing that have long been used to locate lines (and other parametric curves) in noisy images.

Hough transform

The Hough transform (HT) is an image processing technique that is used to detect lines (or

other shapes) in digital images. In a typical use case scenario, the image is preprocessed first (e.g with an edge detector) to obtain pixels that are on the lines (or curves) in image space. Unfortunately, because of noise in the image, there may be missing pixels, and the extracted pixels often do not align well with the lines (curves) we look for. For this reason, it is often nontrivial to group the extracted pixels to an appropriate set of lines (curves). The purpose of the Hough transform is to assign these pixels to line (curve) instances by performing a voting procedure in a parameter space where the lines (curves) can be described with a small number of parameters.

Line detection example

The simplest case of the Hough transform is detecting straight lines. In the image space, a line *l* can be described as

$$y = mx + b$$

where *m* is the slope parameter, and *b* is the intercept parameter. In the Hough transform, the main idea is to represent the line not as a series of image points, but instead, in terms of its parameters. That is, in parameter space, line *l* is represented as a point *(m,b)*.

For practical reasons (namely that *m* and *b* can be unbounded even for small images), we use the polar coordinate system for representing lines in the parameter space:

$$d = x \cos\theta + y \sin\theta$$

where the parameters of the line are *d*, the perpendicular distance of the line from the origin of the image, and *θ*, the angle of the line's normal vector. Notice that both *d* and *θ* are bounded. It is now possible to associate with each line in the image a point *(d,θ)* in parameter space (also called Hough space).

For an arbitrary point (x_0, y_0) in the image, the set of lines that go through this point have parameters *d* and *θ*, such that given the *θ* parameter, *d* is determined by the following formula:

$$d(\theta) = x_0 \cos\theta + y_0 \sin\theta$$

This formula corresponds to a sinusoidal curve in the parameter space, which is unique to pixel (x_0,y_0). If several pixels in the image space are given, we get a separate curve in parameter space for each of them. The points (in parameter space) where these sinusoidal curves cross necessarily correspond to a line that passes through the each of the corresponding image pixels.

That is, the Hough transform converts the problem of detecting collinear points in image space to a problem of detecting intersecting curves in parameter space.

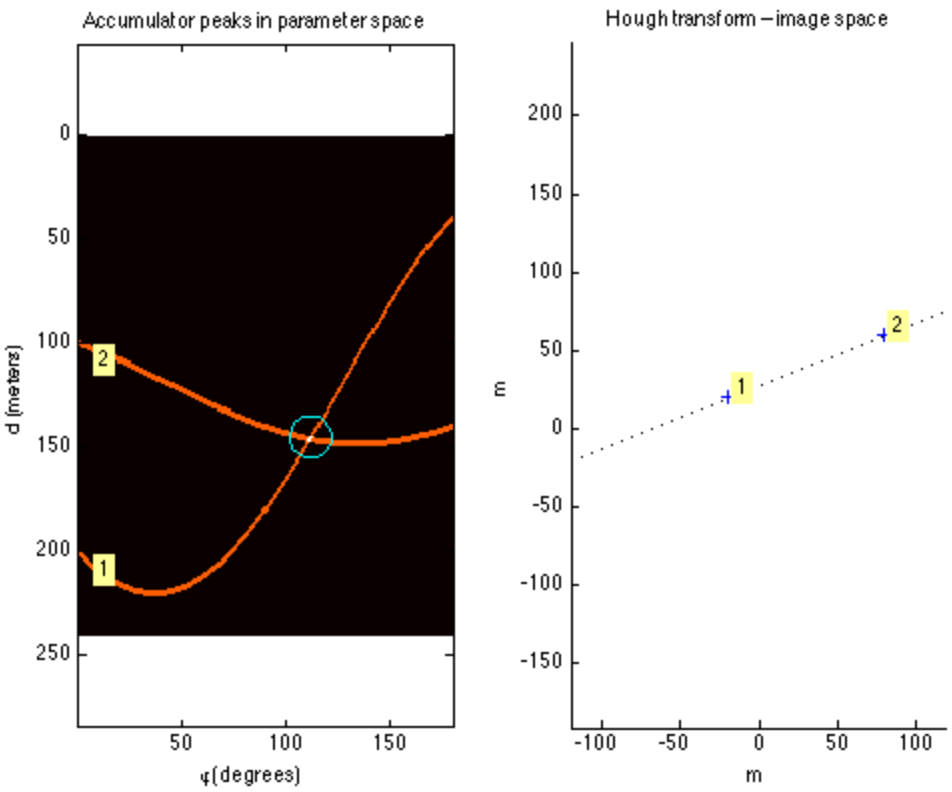

Figure 1: Hough transform for line detection. The two curves in the parameter space (left) correspond to the two points in the image space. The *(d,θ)* parameters corresponding to the intersection point of the two curves (left) define the dashed line (right) that embeds both points.

In practice, the Hough transform implementation uses a two dimensional accumulator matrix, where columns correspond to *θ* bins and rows correspond to *d* bins, with some predefined resolution. Initially, the entire accumulator matrix is set to zero. Then, for each pixel in image space, the algorithm calculates the *d* and *θ* parameters of all lines passing through it, and increases the value of bins corresponding to those *(d,θ)* pairs by one. Once all pixels are processed, the algorithm is looking for local maxima in the accumulator matrix. The *d* and *θ* parameters of each local maxima represent a line in the image, and the corresponding accumulator value tells us how many pixels are positioned on that particular line.

Trajectory estimation with the HT approach

In this section, we discuss a Hough transform based approach to estimating the trajectory of multiple projectiles fired simultaneously.

For simplicity, we restrict the trajectory estimation problem to two dimensions, and assume

that the bullet's deceleration over the sensor field is zero. With these assumptions, the wavefront of the acoustic shock wave consists of two lines, on both sides of the trajectory, where the angle between the wave front and the trajectory is $sin^{-1}(c/v_{bullet})$, c being the known speed of sound, and v_{bullet} is the (unknown but constant) velocity of the projectile.

The sensors report the angle of arrival (AOA) and time of arrival (TOA) of the acoustic shock waves detected, but the association of detections to shots is unknown. The sensor positions are assumed to be known exactly.

The trajectory estimation procedure works as follows:
- First, we preprocess the AOA and TOA detections to compute points of the shock wave fronts at time t_0, where t is an arbitrary time instance that is greater than all the TOA detections;
- second, we use the HT to identify the two sides of shock wave fronts separately, and
- third, we pair them up in a postprocessing step to identify the trajectory.

Preprocessing

If sensor M_i located at (x_i,y_i) detected a shock wave of the j^{th} shot at time $t_{i,j}$, we known that the point $M_i(x_i,y_i)$ is on the line corresponding to one side of the shock wave front at time $t_{i,j}$. From here, given the direction of arrival $n_{i,j}$ ($n_{i,j}$ being a normal vector of unit length to the shock wave front), we can compute a point $P_{i,j}$ that is on the line corresponding to one side of the shock wave front at time t_0:

$$P_{i,j} = M_i + c(t_{i,j} - t_0)\, n_{i,j}$$

where c is the speed of sound. In the preprocessing stage, we compute, for all detections, the points $P_{i,j}$ for an arbitrary time $t_0 = \max(t_{i,j}) + \varepsilon$, where ε is a small positive number.

Hough transform

From here, we execute the classical version of the Hough transform algorithm (described above in the line detection example) on the points $P_{i,j}$, returning the d and θ parameters of those local maxima that have an accumulator value above a user-defined threshold, e.g. 3. That is, result of the algorithm is a set of lines that pass through at by at least three of the input points. We also mark which sensor detections support which line, keeping track of the association between the detections and shock wave front lines.

We can improve on the computational complexity of this approach if we notice that we have, for every point $P_{i,j}$, another piece of information available: $n_{i,j}$, the angle of arrival of the shockwave, which is the normal vector of the line of the shockwave front. The angle of arrival information can be used to restrict the possible values of the φ parameter. (Recall that the φ parameter is the angle of the line's normal vector.) Instead of computing the d values for

$\varphi=0..2\pi$, we only compute it for the interval around the angle of the normal vector $n_{i,j}$:

$$\left[tan^{-1}\frac{y_{n_{i,j}}}{y_{n_{i,j}}} - \varepsilon, tan^{-1}\frac{y_{n_{i,j}}}{y_{n_{i,j}}} + \varepsilon \right]$$

While the classical version of the Hough transform maps a point in image space to a sinusoidal curve in parameter space, by applying the above restriction the point is mapped to a short section of that curve in parameter space. As a result, every non-zero element in the accumulator matrix will represent a valid shockwave front line that is supported by at least one sensor measurement.

Postprocessing

With the HT algorithm, we have identified lines that correspond to one side of the shock wave front, however, we do not know which pairs of such lines correspond to the same shot. To check which pairs of lines constitute feasible shock wave fronts, we carry out a simple geometric check on all *n(n-1)/2* possible pairs of lines.

For a pair of shock wave front lines, we first compute a trajectory candidate: the bisector of the angle at the intersection of the two wave front lines. For this to be a feasible trajectory, we must ensure that the angle of arrival detections that are associated with both shock wave front lines point toward the trajectory candidate. If this condition is true, the trajectory candidate is accepted as a solution. The angle β between the trajectory candidate and the shockwave front line can be used to compute the speed of the bullet over the sensor field as follows:

$$v_{bullet} = c/sin\beta$$

Implementation

The MATLAB implementation of the above described HT based trajectory estimation technique includes the following files:

swtoaaoaloc_ht.m: Find the trajectories for multiple shots fired simultaneously using shockwave time-of-arrival and angle-of-arrival measurements.

Required parameters:

sensor_pos_l:	ith row contains the (x,y) position of the sensors corresponding to the ith detection
sw_aoa_l:	ith row contains the unit vector representing the direction of arrival of the shockwave detected by the sensor at position sensor_pos_l(i,:)
sw_toa_l:	ith row contains the time of arrival of the shockwave detected by the sensor at position sensor_pos_l(i,:)
c:	speed of sound

Optional parameters:

'd_res':	resolution of d
'phi_res':	resolution of phi
'peak_thres':	peak threshold, i.e. the number of points required in order to consider a line a solution
'missdist_thres':	maximum miss distance detection range
'bullet_speed_min':	minimum bullet speed
'bullet_speed_max':	maximum bullet speed

Returns the (x,y,theta) parameters of possibly more than one trajectories, and the detections supporting the respective solutions:

traj_l:	the ith row contains the (x,y) coordinates of a point on the trajectory, and theta, the trajectory angle
supporter_map_l:	the element at (i,j) is 1 if the ith solution is supported by the jth detection, otherwise 0

Operation:

The function first computes the points on the shock wave fronts at time t_0, i.e. moves the sensor positions in the direction opposite the shock wave angle of arrival by $c(t_{i,j} - t_0)$, where $t_{i,j}$ is the time of arrival and c is the speed of sound. Then, it calls

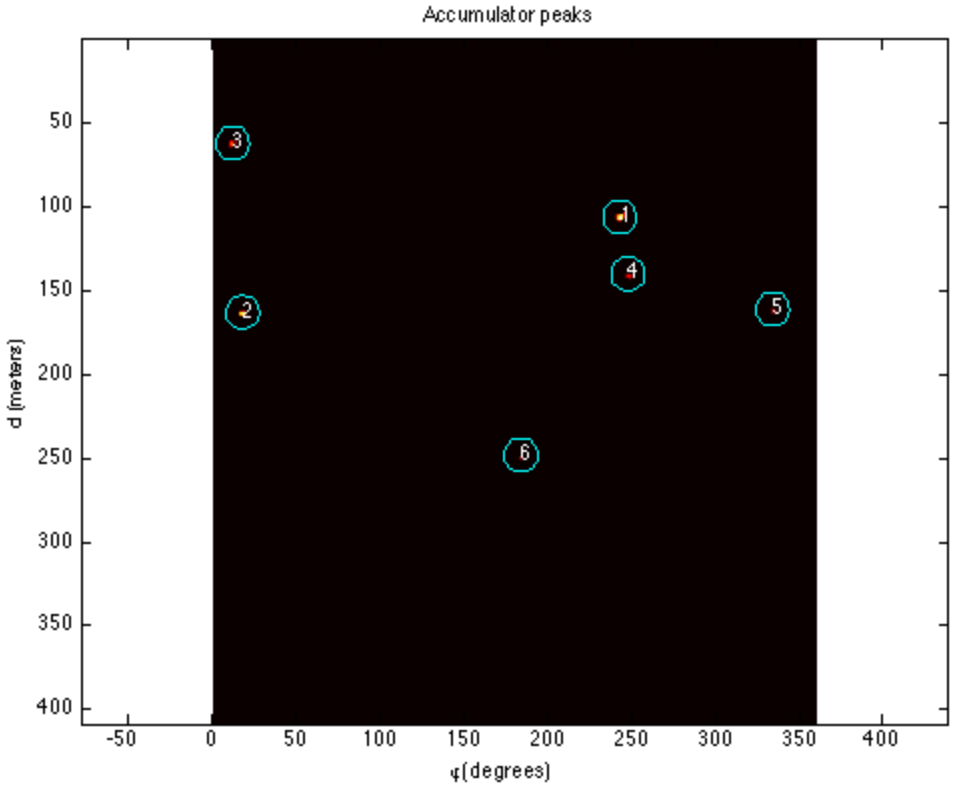

Figure 2: Accumulator values. Lighter values indicate higher accumulator value. There are 6 peaks identified in the parameter space, representing 6 distinct shockwave front lines.

point_with_normal_hough_transform to find the shock wave front lines. Finally, for all possible pairs of shock wave front lines, a trajectory candidate is computed. A simple test is used to check whether the trajectory candidate is a feasible trajectory. Given the trajectory candidate, for all AOA detection that support either of the wave front lines, we compute the shockwave cone angle. We check if the computed angle is within the range computed from the parameters 'bullet_speed_min' and 'bullet_speed_max' (using the relation between the cone angle and the bullet speed). The trajectory candidate is accepted if this condition holds, and rejected otherwise.

point_with_normal_hough_transform.m: Solve for plane wave fronts given some points on the wave front using the Hough transform

Given a list of points and corresponding normal vectors, find the lines (wave fronts) on which these points lie and have the same normal vectors, using the Hough transform.

The HT Parameter space is 2-dimensional: d is the perpendicular distance of the origin from the line, phi is the angle of the normal vector.

Equation of line:

$$x*\cos(phi)+y*\sin(phi) = d$$

or

$$y = -(x*\cos(phi)-d)/\sin(phi)$$

Required parameters:

p_l:	points on the wave front with normal vector angles (x,y coordinates and angle in each row)
d_range:	range of d parameter [d_min, d_max]

Optional parameters:

'd_res':	resolution of d
'phi_res':	resolution of phi
'peak_thres':	peak threshold, i.e. the number of points required in order to consider a line a solution

Returns the (d,phi) parameters of possibly more than one shocwave wavefronts, and the detections supporting the respective solutions:

line_l: the ith row contains the d and phi parameters of the ith solution

supporters: the element at (i,j) is 1 if the ith solution is supported by the jth detection (i.e. the jth point is on the ith line)

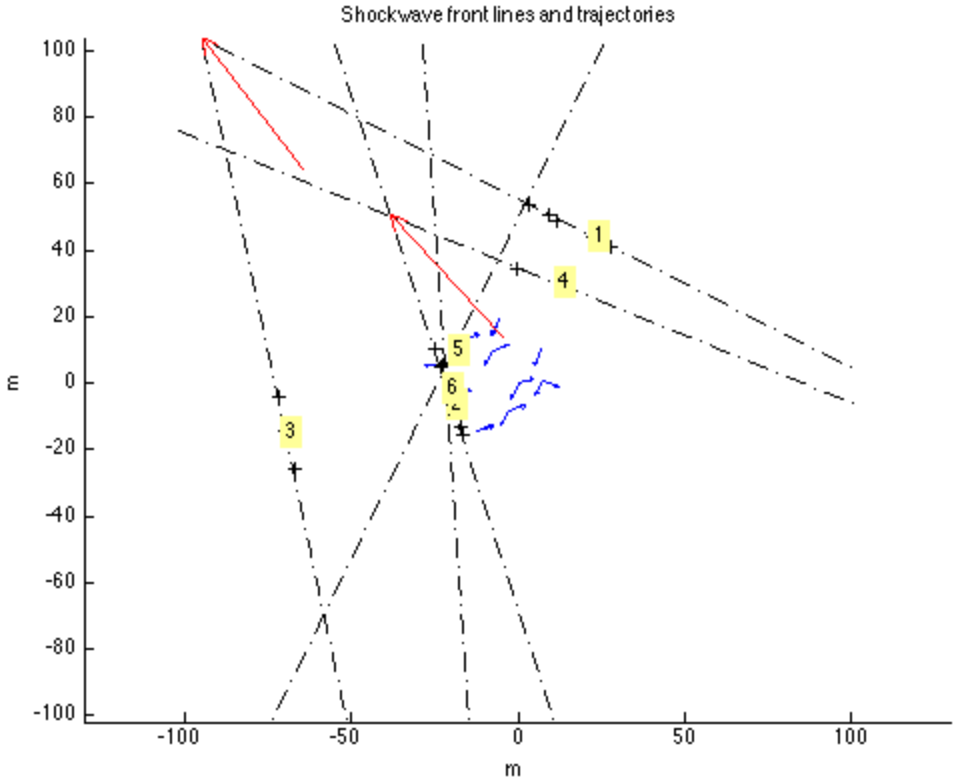

Figure 3: 6 shockwave front lines are identified by the Hough transform (dashed lines). Red arrows represent the valid trajectory estimates. Short blue arrows represent the shockwave AOA measurements.

A 2-dimensional accumulator matrix is initialized to all zeros, quantizing the parameter space according to the 'd_res' and 'phi_res' parameters. For all points in p_l (containing the x,y coordinates and the normal vector angle), the d parameter is computed for three φ bins: the φ bin corresponding to the normal vector angle of the point, and the two neighboring bins. Then, the houghpeaks function (MATLAB image processing toolbox) is invoked to locate the peaks in the accumulator matrix, the parameters of which represent the shockwave front lines identified. Finally, for all lines and for all points we check if the point supports the line (i.e. that the x,y coordinates of the point are sufficiently close to the line, and the point's normal vector angle is sufficiently close to the normal vector of the line), and return the results in the 'supporters' boolean matrix variable.

swtoaaoaloc_ht_test_aberdeen.m: Test script that runs the HT based trajectory

estimator on the Aberdeen dataset

The Aberdeen data set was collected by NIST at an independent evaluation of Vanderbilt's wireless sensor network based countersniper system at the US Army Aberdeen Test Center in April 2006. The experiment was setup on a shooting range with mock-up wooden buildings and walls for supporting elevated shooter positions and generating multipath effects. Ten 4-channel acoustic sensor nodes were deployed on surveyed points in an approximately 30×30 m area. The data set contains 196 shots. There were five fixed targets behind the sensor network.

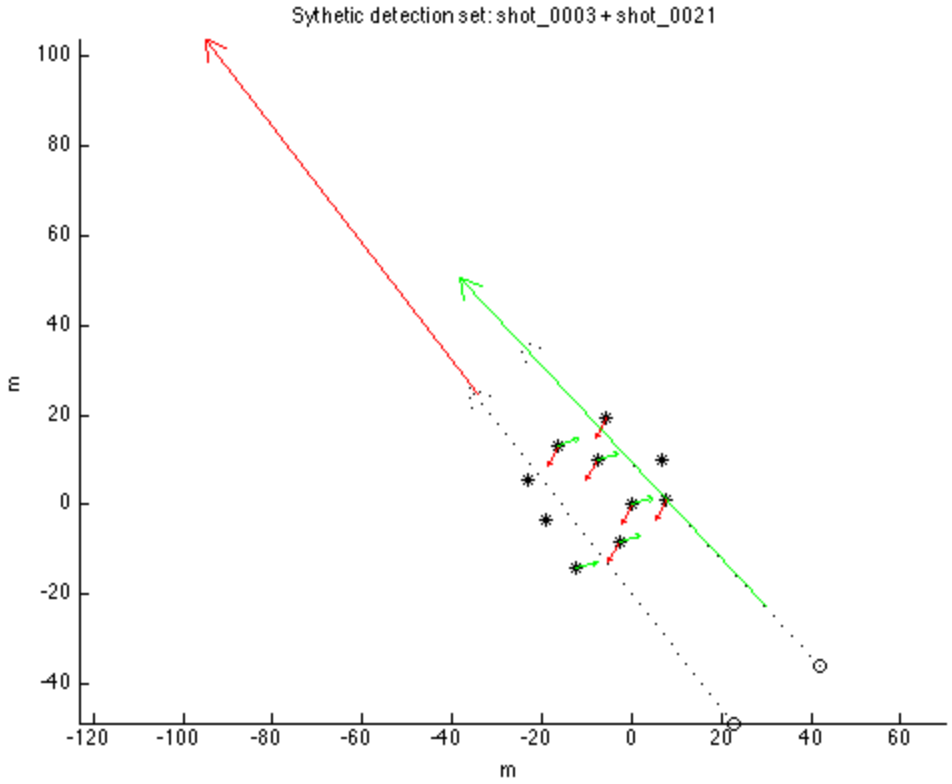

Figure 4: Trajectories calculated for a 2-shot synthetic data set. The sensor positions are represented stars, the shooter positions by small circles. The dashed lines represent the ground truth trajectores. The long green and red arrows are the computed trajectories. The short green and red arrows represent the shockwave AOA detections associated with the green or red trajectory estimates, for the respective sensors.

Several firing positions were located at each of the firing lines at 50, 100, 200 and 300 meters. Six different weapons were utilized: AK47 and M240 firing 7.62 mm projectiles, M16, M4 and M249 with 5.56mm ammunition and the .50 caliber M107. The sensors remained static during the test. The positions of sensors, shooters and targets were surveyed with high precision DGPS. The sensor network was time synchronized with better than 100 microsecond accuracy between any two nodes. The mean AOA accuracy was under 1 degree.

The test script operates as follows. First, the data set is loaded, and the sensor, shooter and target positions are converted to an East-North-Up 3-dimensional Cartesian coordinate system. Then, two shots are selected, and a synthetic detection set is generated by adjusting and the detection times of the later shot such that the two shots become almost simultaneous: the detection times are on average 100 milliseconds apart. The swtoaaoaloc_ht function is called with the synthetic detection set as input. Finally, the computed trajectories are plotted, along with the ground truth for reference purposes (true trajectory is represented with a dashed line). For each computed trajectory, we plot the AOA detections (vectors of unit length pointing towards the shockwave front) with the same color as the trajectory itself, to visualize which association between the detections and the trajectory estimate.

Shooter position estimation

In this section, we discuss a consistency function based approach, inspired by the Hough transform, to estimating the position of multiple shooters assuming that the weapons are fired simultaneously.

The sensors report the angle of arrival (AOA) and time of arrival (TOA) of the acoustic muzzle blasts detected, but the association of detections to shots is unknown. The sensor positions are assumed to be known exactly.

Hough transform based approach

A Hough transformation based approach can be applied to this problem as follows. Let us assume, for simplicity, that we search for the shooter positions in a 2-dimensional space. In this case, we set up the parameter space as *(x,y,t)*, where *x* and *y* are the coordinates of the position of the shooter, and *t* is the time of the shot. The parameter space is necessarily bounded by the muzzle blast detection range of the sensors in the (x,y) spatial dimensions. The temporal dimension is also bounded by the time of the latest/earliest detections, plus/minus the time it takes for the sound to travel across the spatial dimensions of the parameter space.

How do we map muzzle blast detections to this parameter space? Intuitively, if the shot happened 2 seconds before the sensor detected the muzzle blast, the shooter must be on a circle centered at the sensor's position with radius $r = 2c \approx 680m$, where c is the speed of sound. Similarly, if the shot was fired 1 second before the detection, the radius of the circle would be $r = c \approx 340m$. If the shot was fired exactly at the time of the detection, the radius of the circle is 0 (and the shooter position is exactly at the position of the sensor).

By generalizing this, if sensor M_i located at (x_i,y_i) detected a muzzle blast of the j^{th} shot at time $t_{i,j}$, we known that the point representing the shooter position and shot time (x_j,y_j,t_j) in parameter space is on the surface of a cone. The axis of the cone is perpendicular to the (x,y) plane, tip of the cone is at $(x_i,y_i,t_{i,j})$, and the cone angle is $\tan^{-1}c$, c being the speed of sound.

If a shot happened at position (x_j,y_j) at time t_j, the cone surfaces corresponding to detections of sensors that observed the muzzle blast event necessarily intersect at point (x_j,y_j,t_j). It is easy to see that at least three sensor detections are required to unambiguously identify the shooter position.

Therefore, the straightforward Hough transform based solution to this problem works as follows:
- A 3-dimensional accumulator matrix is initialized to all zeros.
- For each time-of-arrival detection $t_{i,j}$ by sensor M_i located at (x_i,y_i), the above conical surface is computed, and the accumulator bins that the cone surface intersects are incremented.
- A peak detector is invoked on the accumulator matrix that extracts the parameters of the local maxima (above some reasonable threshold of 3 or more). The (x_j,y_j) coordinates of the maxima are the shooter position, and t_j is the time of shot.

The above technique can easily be augmented by angle of arrival information. When computing the cone surface, only those accumulator bins are incremented that are (approximately) at a given bearing from the cone's axis. That is, if the sensor position is (x_i,y_i), the angle of arrival is γ, and we computed that the point (x_k,y_k,t_k) is on the cone surface, we only increment the accumulator bin corresponding to this point if it is in the direction of the AOA. Formally:

$$\gamma - \epsilon < tan^{-1}\frac{y_k - y_i}{x_k - x_i} < \gamma - \varepsilon$$

where ε is a suitable constant to accommodate the handling of AOA measurement errors.

Implementation

Our MATLAB implementation of the Hough transform inspired shooter position estimation technique uses a sparse representation of the accumulator matrix, and iterative refinement that decreases the bin size on demand to avoid heavy memory usage. While this technique is conceptually equivalent to the above described one, the structure of the code differs significantly.

mbtoaaoa_loc.m: Find the location of one or more shooters using muzzle blast angle of arrival and time of arrival detections from multiple sensors.

Required parameters:

cell_l: list of accumulator bins: ith row contains the (x,y,t) coordinates of the bins,
bin size (as a fraction of C_0) and the list of indexes of the sensors that support that bin

C_0: dimensions of the initial cell

mic_l: list of sensor positions (x,y)
toa_l: ith row contains the time of arrival of the muzzle blast detected by the sensor at position mic_l(i,:)
aoa_l: ith row contains the unit vector representing the direction of arrival of the muzzle blast detected by the sensor at position mic_l(i,:)
aoa_accuracy: accuracy of angle of arrival measurements (radians)
c: speed of sound

Optional parameters:
'mb_cutoff': muzzle blast detection range of the sensors
'zoom_factor': the bins will be subdivided into zoom_factor^3 equal smaller bins in each iteration (to refine the results)
'supporter_thres': supporter threshold, i.e. the number of supporters required in order to consider the bin's (x,y,t) parameters a solution
'iteration_thres': maximum number of refinements
'size_thres': minimum allowed size of a bin - bins with any dimension smaller than size_thres will not be subdivided

Returns:
soln_l: the list of bins (x,y,t), bin size (as a fraction of C_0) and the indexes of the sensors that support the bin. (x,y) are the coordinates of the shooter, t is the shot time.
C_0: the initial cell size
elim_soln_l: the list of eliminated cells in the last iteration

Operation:
If it is the C_0 parameter is an empty matrix, The function first sets up the initial cell to cover the joint sensing range of the sensors and all feasible shot times. If the cell_l parameter is empty, cell_l is set to C_0.

In an iteration, for every cell in cell_l, the the cell is subdivided into zoom_factor^3 child cells. We test which sensor detections agree with the shot being fired from the position and time the cell represents by calling mbtoaaoa_supporters. If the number of supporting sensors is less than supporter_thres, the cell is discarded, otherwise it is further subdivided in the next iteration.

The iteration stops if any of the following conditions are met:
- All cells have been discarded: there is no solution.
- Iteration threshold is reached or the cell size has reached size_thres: there is no need to increase the granularity of the solution any more.

The solutions (list of cells and their supporters) are returned in soln_l. The solutions eliminated in the last iteration are returned in elim_soln_l.

mbtoaaoa_supporters.m: Identify the sensor detections supporting a bin.

Required parameters:

mics:	list of sensor positions (x,y)
toas:	ith row contains the time of arrival of the muzzle blast detected by the sensor at position mic_l(i,:)
aoas:	ith row contains the unit vector representing the direction of arrival of the muzzle blast detected by the sensor at position mic_l(i,:)
aoa_accuracy:	accuracy of angle of arrival measurements (radians)
c:	speed of sound
cell:	row vector containing (x,y,t) coordinates of the cell, cell size as a fraction of C_0, supporter indexes
cell_size:	size of the cell in x,y and t dimensions

Returns:

cell:	row vector containing (x,y,t) coordinates of the cell, cell size as a fraction of C_0, updated supporter indexes
supporter_cnt:	number of sensor detections that are consistent with a shot fired from coordinates (x,y,t) at the cell's center

Operation:

For each sensor, the function checks if the conical surface a sensor detection defines passes through the cell. If this condition holds, the supporter_cnt value is incremented, and the sensor's index is included in the cell's row vector.

mbtoaaoa_loc_find_max_supporter_threshold.m: Find the highest supporter threshold that still yields a solution.

Required parameters:

soln_l:	list of accumulator bins: ith row contains the (x,y,t) coordinates of the bins, bin size (as a fraction of C_0) and the list of indexes of the sensors that support that bin
C_0:	dimensions of the initial cell
mic_l:	list of sensor positions (x,y)
toa_l:	ith row contains the time of arrival of the muzzle blast detected by the sensor at position mic_l(i,:)
aoa_l:	ith row contains the unit vector representing the direction

	of arrival of the muzzle blast detected by the sensor at position mic_l(i,:)
aoa_accuracy:	accuracy of angle of arrival measurements (radians)
c:	speed of sound

Optional parameters:

'mb_cutoff':	muzzle blast detection range of the sensors
'zoom_factor':	the bins will be subdivided into zoom_factor^3 equal smaller bins in each iteration (to refine the results)
'size_thres':	minimum allowed size of a bin - bins with any dimension smaller than size_thres will not be subdivided

Returns:

best_supporter_thres:	the highest supporter threshold that still yields a shooter position solution
best_input_l:	the list of fine-grained bins that can be used as an input to mbaoa_loc and will produce a non-empty result
best_soln_l:	list of solution bins for the best supporter threshold
C_0:	the initial cell size

Operation:
If it is the C_0 parameter is an empty matrix, The function first sets up the initial cell to cover the joint sensing range of the sensors and all feasible shot times. If the cell_l parameter is empty, cell_l is set to C_0.

This function iteratively calls mbtoaaoa_loc, increasing the supporter_thres parameter in every iteration. The iteration stops if when mbtoaaoa_loc returns with an empty bin list. The function returns supporter_thres-1, the highest threshold that still yielded a result.

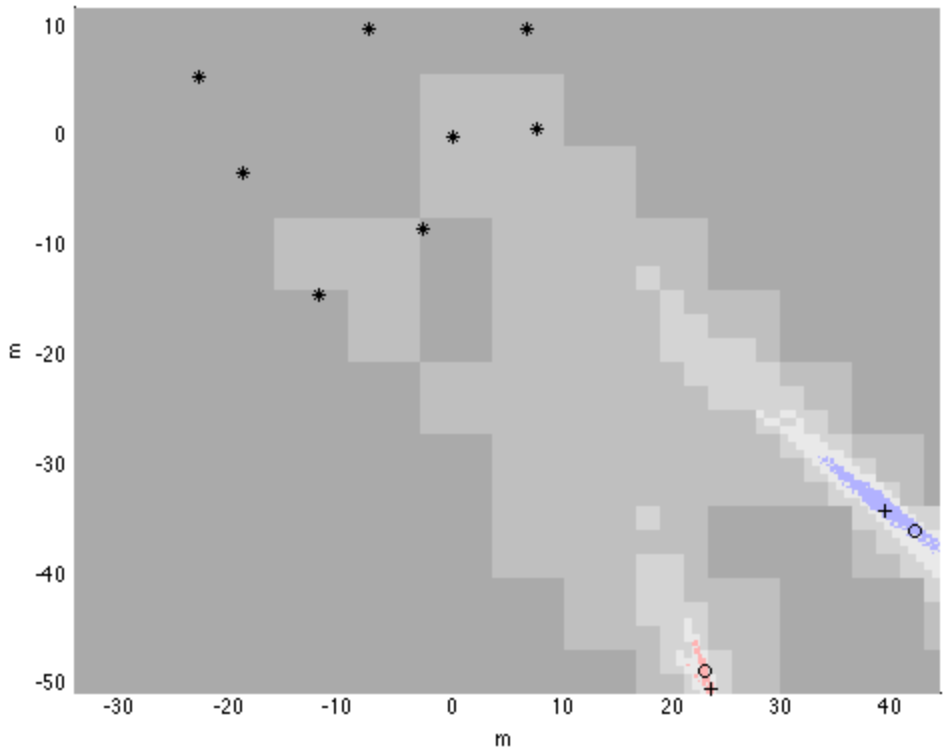

Figure 5: Shooter location result. The sensor positions are represented by black asterisks, the ground truth shooter positions are represented by small black circles. The color of the bins encodes the supporter counts - lighter colors meaning more supporters. The bins plotted in blue and red are the peaks with the highest supporter counts. The estimated shooter positions are marked with + signs.

mbtoaaoaloc_ht_test_aberdeen.m: Test script that runs the HT based shooter position estimator on the Aberdeen dataset

The test script operates as follows. First, the data set is loaded, and the sensor, shooter and target positions are converted to an East-North-Up 3-dimensional Cartesian coordinate system. Then, two shots are selected, and a synthetic detection set is generated by adjusting and the detection times of the later shot such that the two shots become almost simultaneous: the detection times are on average 100 milliseconds apart.

The mbtoaaoa_loc_find_max_supporter_threshold function is called with the synthetic detection set as input. It returns the highest supporter threshold for which there exists a shooter position solution. The results are further refined by mbtoaaoa_loc using this threshold. Since potentially a large number of bins are returned, we resort to a k-means clustering to cluster the results, and use the point of mass of the clusters as a solution. Finally, the computed shooter positions are plotted, along with the ground truth for

reference purposes.

Conclusion

In this report, we introduced Hough transformation based solutions for the trajectory estimation and for the shooter location problems where multiple simultaneous shots necessitate resolving the data associations between acoustic event detections and the corresponding shots. We implemented the above described algorithms in MATLAB and provided preliminary results using a data set of time-synchronized, precisely localized network of multi-channel sensors detecting angle of arrivals and time of arrivals of shock wave and muzzle blast detections.

Acknowledgement

This work was sponsored in part by the Army Research Office. Their support is gratefully acknowledged.