Conference on Mathematics and Applications of Data/Image Coding, Compression, and Encryption IV
SPIE's International Symposium on Optical Science and Technology, San Diego, CA, July 29 – August 3, 2001

Data Compression via Pulse-to-Pulse Redundancy for Radar Emitter Location*

Mark L. Fowler† and Zhen Zhou

Department of Electrical Engineering
State University of New York at Binghamton
Binghamton, NY 13902

ABSTRACT

An effective method for geolocation of a radar emitter is to intercept its signal at multiple platforms and share the data to allow measurement of the time-difference-of-arrival (TDOA) and the frequency-difference-of-arrival (FDOA). This requires effective data compression. For radar location we show that it is possible to exploit pulse-to-pulse redundancy.

A compression method is developed that exploits the singular value decomposition (SVD) to compress the intercepted radar pulse train. This method consists of five steps: (i) pulse gating, (ii) pulse alignment, (iii) matrix formation, (iv) SVD-based rank reduction, and (v) encoding. Matrix formation places aligned pulses into rows to form a matrix that has rank close to one and SVD truncation gives a low rank approximate matrix. We show that (i) compression is maximized if the matrix is made to have two-thirds as many rows as columns and (ii) truncation to a rank-one matrix is feasible. We interpret this as extracting a prototype "pulse trainlet."

The maximum compression ratio is expressed in terms of the number of pulses and the number of samples per pulse and point out a particularly interesting and important characteristic – the compression ratio increases as the total number of signal samples increases. Theoretical and simulation results show that this approach provides a compression ratio up to about 30:1 in practical signal scenarios.

Keywords: data compression, singular value decomposition, emitter location, time-difference-of-arrival, TDOA, frequency-difference-of-arrival, FDOA

1. INTRODUCTION

A common way to locate electromagnetic emitters is to measure the time-difference-of-arrival (TDOA) and the frequency-difference-of-arrival (FDOA) between pairs of signals received at geographically separated platforms.[1,2,3] The measurement of TDOA/FDOA between these signals is done by coherently cross-correlating the signal pairs.[2,3] This requires that the signal samples of the two signals are available at a common platform, which is accomplished by transferring the signal samples over a data link from one platform to the other.

An important aspect of this that is not widely addressed in the literature is that the available data link rate often is insufficient to accomplish the transfer within the time requirement unless some form of lossy data compression is employed. To mitigate this, various data compression approaches have been proposed,[4–9] although they have not been designed to fully exploit the characteristics of radar signals. For the case of white Gaussian signals and noises, Matthiesen and Miller[4] established bounds on the rate-distortion performance for the TDOA/FDOA problem and compared them to the performance achievable using scalar quantizers, where distortion is measured in terms of lost SNR due to the mean square error (MSE) of lossy compression. However, these results are not applicable when locating radar emitters because the signals encountered are not Gaussian. A method using block adaptive scalar quantization was proposed[5] and analyzed[6] to show that it was marginally effective for various signal types. Wavelet-based methods have been proposed[7] and demonstrated[8] to give compression ratios on the order of 6 to 7 for some radar signals. A method that optimally trades between decimation and quantization has been developed for non-radar-like signals and shown to perform better than either method alone.[9] However, none of these previously proposed methods exploits the inherent pulse-to-pulse redundancy characteristic of most radar signals.

For the radar case, the fact that (at least) one platform must be operating at an SNR high enough to detect pulses can be exploited as a first step towards reducing the transferal by not sending the samples between detected pulses (i.e., gating).

* This work was supported by the Lockheed Martin Corporation under Contract UA-199865.
† Correspondence: mfowler@binghamton.edu

However, even with this reduction due to gating, the transferal time is still excessive given the rates for current and projected data links. In the method proposed here, once the pulses have been gated they are formed into a matrix in such a way that the resulting matrix has an effective rank of one, due to the redundancy between pulses. Then the singular value decomposition (SVD) is used to exploit this redundancy to achieve significant levels of compression ratio that exceed what is possible using the previously proposed general methods.[4-9]

The two signals to be correlated are the complex envelopes of the received RF signals. The two noisy received signals to be processed are notated as

$$\begin{aligned} \hat{s}(k) &= s(k) + n(k) \\ \hat{d}(k) &= d(k) + v(k) \end{aligned} \tag{1}$$

where $s(k)$ and $d(k)$ are the complex baseband signals of interest and $n(k)$ and $v(k)$ are complex white Gaussian noises. The signal $d(k)$ is a delayed and doppler shifted version of $s(k)$. The signal-to-noise ratios (SNR) for these two signals are denoted *SNR* and *DNR*, respectively[‡]. To cross correlate these two signals one of them (assumed to be $\hat{s}(k)$ here) is compressed, transferred to the other platform, and then decompressed before cross-correlation, as shown in Figure 1. Signal $\hat{s}_c(k)$ has SNR of SNR_c after lossy compression/decompression.

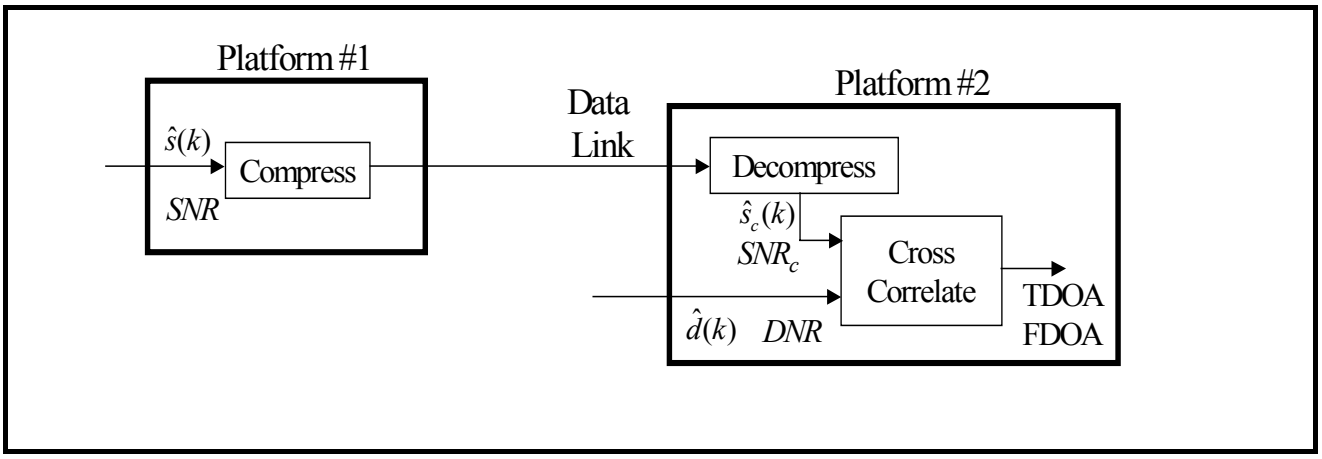

Figure 1: System Configuration for Compression

2. FORMING THE PULSE MATRIX

1. Pulse Gating and The Unaligned Pulse Matrix

The emitter location system consists of three or more platforms, each outfitted with identical receiving and processing equipment. Once signal data is collected at all of the platforms, the SNR is estimated at each platform and the one with the highest SNR is chosen as the one to transmit its data to the others for subsequent correlation processing; this platform is called the transmitting platform (Tx platform) and the other platforms are called the correlating platforms (Cx platforms). The Tx platform is required to be at a high enough SNR to allow standard radar intercept signal processing to be done.[10,11] Because modern radars can change modes we assume that preliminary subtrain-extraction (de-interleaving) processing has grouped the signal of interest into one or more subtrains, each having pulses from the same mode of operation – such processing is a standard part of typical electronic warfare systems (this processing also removes pulses from other emitters).[10,11] Here we consider compressing one such subtrain.

As part of this interception processing, the Tx platform detects the individual pulses of the emitter of interest, gates around them, and keeps only the signal samples that lie inside the pulse gates; the numbers of samples removed between the pulses are also kept as side information. This process is called pulse gating and is illustrated in Figure 2, where for clarity the signal is shown only as a real-valued envelope. The gated pulse train is compressed (as described below) and transmitted

[‡] SNR (non-italic) represents an acronym for signal-to-noise ratio; *SNR* (italic) represents the SNR for $\hat{s}(k)$.

(along with the side information) over the data link to the Cx platforms. At each Cx platform receiving this broadcast data, the data is decompressed to reconstruct the gated pulse train, zeros are inserted between pulses according to the side information N_1, N_2, etc., and then it is cross correlated with the signal received locally at the Cx platform to allow estimation of the TDOA/FDOA values. The sets of TDOA/FDOA estimates from the various Cx platforms are then combined to estimate the emitter location.[1]

After pulse gating at the Tx platform, as a first step of the compression the pulses are put into a matrix (called the nonaligned pulse matrix) with one pulse per row, as is shown in Figure 2. This matrix would have rank one if: (i) there were no noise or propagation effects, (ii) the pulses were perfectly time aligned – i.e. perfect gating, and (iii) the radar's pulse repetition interval (PRI) were constant and an integer multiple of the sampling interval T. Because a $J\times K$ rank one matrix can be completely expressed using only $J + K$ numbers rather than JK numbers, this is the idea behind the method proposed here. Unfortunately all of these effects are present and they each act to increase the rank of the pulse matrix. However, all but the first of these causes of increased rank can be mitigated by performing time alignment on the pulses in the nonaligned pulse matrix.

2. Pulse Alignment

The goal of time alignment is to transform the original pulse matrix into a matrix that is as close to rank one as possible. Because the radar's PRI and the platform's sampling interval T are incommensurate (i.e., there is no *integer k* such that PRI = kT) the needed time alignment is not an integer number of samples. Therefore we need a method of shifting pulses by a fraction of a sampling interval. There are several ways to accomplish a fractional shift and we assess their performance here for this application. The amount of shift that each pulse needs is determined by cross correlating the pulses in the pulse matrix.

Let $\mathbf{P}_{na}$ be the matrix of nonaligned pulses that are extracted by the gating procedure, as shown in Figure 2 where we use a 3-D perspective view of this matrix, where the 3rd dimension illustrates the matrix elements' values. Assume that there are p gated pulses each having n samples per pulse; then the total number of samples is pn and $\mathbf{P}_{na}$ is a $p\times n$ matrix. For alignment processing we view each pulse as existing over the same time interval. In particular, the j^{th} row is considered to be a discrete-time pulse given by $p_j(k)$ for k = 0, 1, … , n–1 and j = 0, 1, … , p–1 so that matrix $\mathbf{P}_{na}$ has its j,k element given by $\mathbf{P}_{na}(j,k) = p_j(k)$. We choose the pulse with the largest energy as the reference pulse, to which all the other pulses will be aligned; let this pulse be denoted as $p_m(k)$. Then to find the time alignment needed for the j^{th} pulse ($j \neq m$) we cross correlate it with the reference pulse to give

$$C_j(k) = \sum_{l=0}^{n-1} p_m(l)\, p_j^*(l-k) \tag{2}$$

and then interpolate $C_j(k)$ to find the location of its peak, which is then taken as the time shift Δ_j to be applied to $p_j(k)$ to align it with $p_m(k)$. The time shift Δ_j can be written as $\Delta_j = D_j + \delta_j$ where D_j is an integer and $0 \le \delta_j < 1$. The integer part of the alignment can be handled separately according to $\tilde{p}_j(k) = p_j(k + D_j)$. Now $\tilde{p}_j(k)$ must be shifted by an amount that is less than a single sample.

There are several ways to impart a fractional delay to a signal. Which one is used depends partly on whether the signal to be delayed is available in its entirety as one block or is available sequentially, either sample-by-sample or on a subblock-by-subblock basis; that is, it depends on the level of latency that is acceptable. Other considerations are accuracy and complexity. Here we consider four different methods and assess them on these merits. The methods are (i) a full-block DFT method based on the delay property of the DFT,[13] (ii) a subblock-based version of the DFT method,[13] (iii) fixed FIR filters designed for fractional delay,[14] and an adaptive FIR filter for fractional delay.[13] Zhou[13] compares the accuracy on the basis of magnitude vs. frequency, delay vs. frequency, and SNR vs. delay value; the results are summarized in Table 1.

Because in this application the signals to be delayed (the individual pulses) can be quite short it is clear that the adaptive FIR method is not suitable. The fixed FIR methods are also not the best choice because (i) the signals can be short, (ii) the signals are wideband so the frequency-specific characteristic is a major disadvantage, and (iii) the worst-case accuracy vs. delay may not be acceptable. That leaves the subblock DFT and the full-block DFT methods. For the current application, where the signals to be fractionally delayed are each individual pulse, there is no penalty in having to wait for an entire pulse to become available because the front-end processing must first identify all the pulses before any subsequent processing is done; furthermore, the required fractional delay can't be determined until the entire pulse is available. Therefore, for this

application we use the full-block DFT approach; for details on the subblock DFT method and the adaptive FIR methods see Zhou.[13]

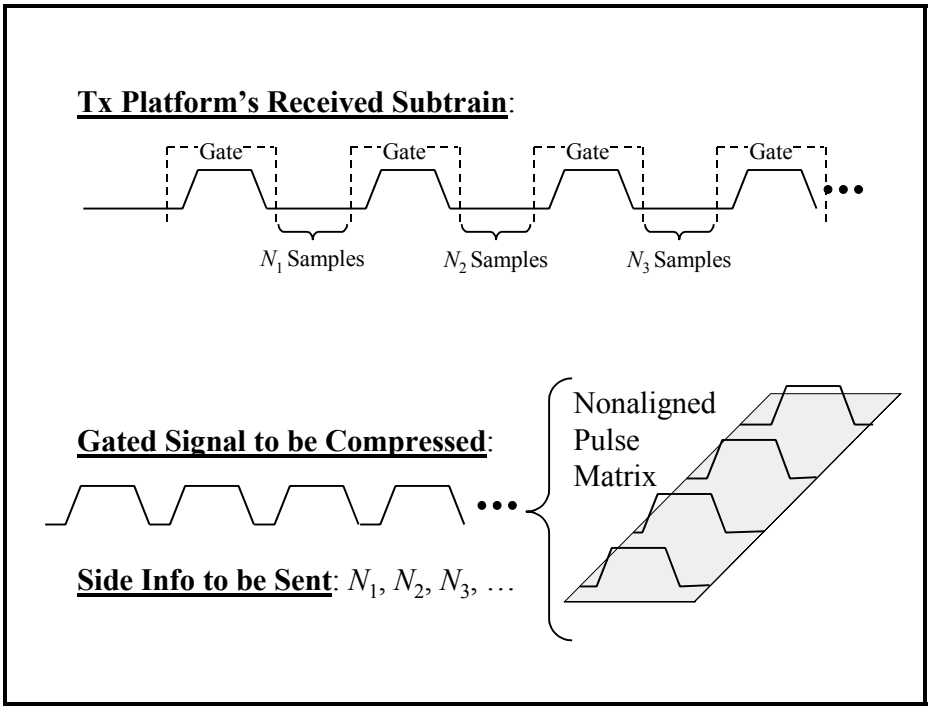

Figure 2: Pulse Gating Before Compression Processing

Table 1: Comparison of Various Fractional Delay Methods

Characteristic	Full-Block DFT	Subblock DFT	Fixed FIR	Adaptive FIR
Accuracy	Very High	High	Low to Medium	High
Latency	Very High	High	Low	Low
Complexity	High	High	Low	Medium
Pros	• Best for Short Signals • Insensitive to delay value • Not Freq. Specific	• Good for Short Signals • Not Freq. Specific	• Maximally flat at zero frequency	• Not frequency specific • Insensitive to delay value • Handles nonstationarity
Cons	• Entire signal block needed	• Blocky Effect	• Freq. Specific • Bad Worst Case vs. Delay • Not best for short signals	• Sensitive to selection of step size • Not suited for short signals

To give a delay of $\delta \in (0,1)$ requires passing the signal through a system with frequency response given by

$$H(\Omega) = e^{-j\Omega\delta}, \quad \Omega \in [-\pi, \pi) \tag{3}$$

where $\Omega = 2\pi fT$ is the discrete-time frequency for a sampling interval of T. This can be implemented using DFT properties as follows:

- Compute the DFT (via the FFT) of the signal using zero-padding to ensure that the time shift is not a circular one;
- Multiply the DFT by $e^{-j\Omega\delta}$ evaluated at the DFT frequencies;
- Compute the inverse DFT of the result (via the FFT).

The effectiveness of the fractional delay method for this application can be seen by assessing its ability to reduce the effective rank of the pulse matrix. The effective rank of a matrix is best assessed via the SVD.[12] Let **P** be the matrix that is obtained from $\mathbf{P}_{na}$ after aligning its pulses as described above. If the alignment method is effective at creating a matrix with effective rank one, then all but the first singular value of **P** should be insignificant. Because the sum of the squares of the singular values gives the energy of the signal it makes more sense to plot the squares of the singular values; for each case, normalizing by the largest singular value improves the comparison between various cases (e.g., nonaligned, aligned, etc.). Figure 3 shows the squares of the normalized singular values as a function of singular value index for the unaligned matrix, the aligned matrix using only integer alignment, and also using fractional alignment; the case shown here is for a simulated linear FM radar signal sampled at an interval that is incommensurate with the PRI. From this we see the effectiveness of the fractional alignment method – the effective rank of the fractionally aligned matrix can be seen to be close to one. This is the basis of the compression method developed here: compression is achieved because the fractionally-aligned matrix can be closely approximated in terms of a rank one matrix.

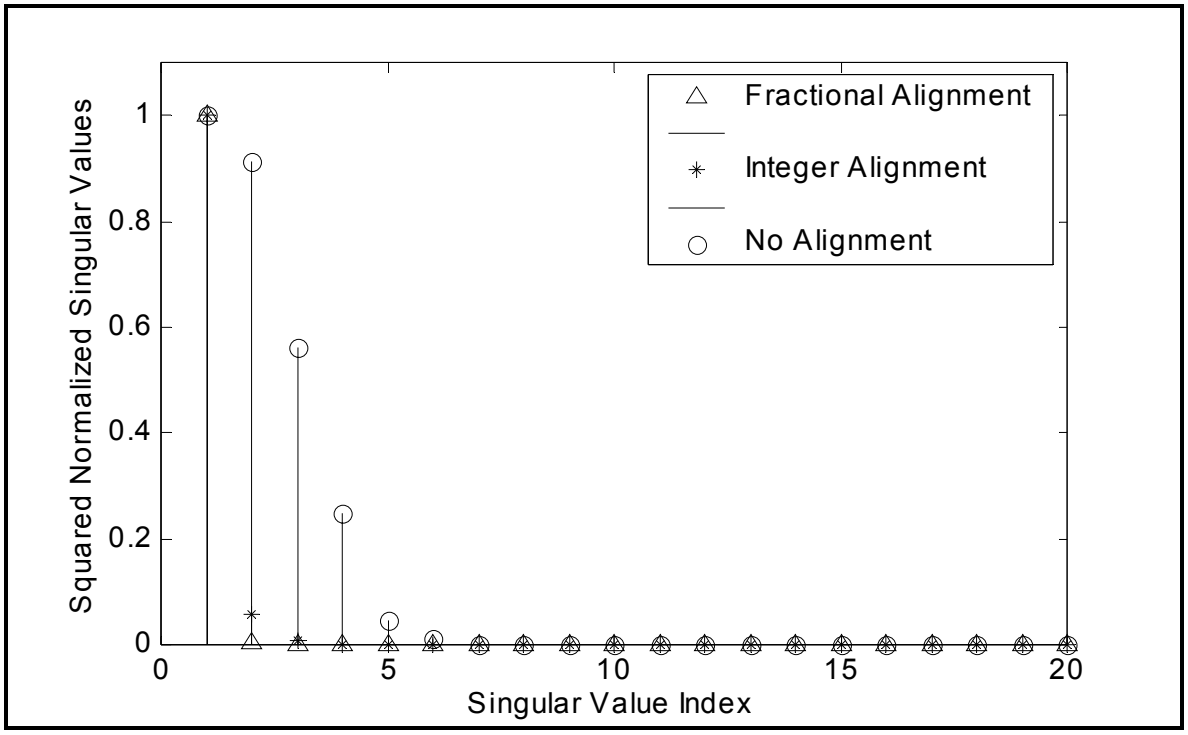

Figure 3: Squared Normalized Singular Values for Aligned and Unaligned Matrices

3. COMPRESSING THE ALIGNED PULSE MATRIX

1. Prototype Pulse Extraction

If we denote the $p \times n$ aligned pulse matrix by **P**, its SVD is

$$\mathbf{P} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^H \,, \tag{4}$$

where r is the rank of **P**, $\mathbf{u}_i$ is the i^{th} left singular vector, $\mathbf{v}_i^H$ is Hermitian transpose of the i^{th} right singular vector, and σ_i is the i^{th} singular value, ordered such that $\sigma_i \geq \sigma_{i+1}$. Each term in the sum in (4) is a rank-one matrix. If we truncate this sum to only $k < r$ terms we get the rank-k matrix

$$\mathbf{P}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^H \tag{5}$$

that best approximates **P** in the sense that the sum of the squares of the elements of **P** - $\mathbf{P}_k$ is smaller than for any other rank-*k* matrix. Note that in our case the matrix contains the pulses and therefore this approximation gives the smallest mean square error (MSE) between the original pulse train and the approximate pulse train formed by concatenating the de-aligned rows of $\mathbf{P}_k$. This minimum MSE property is the basis for using the SVD here.

In the perfect-signal scenario (i.e., where there is no noise and no pulse misalignment) **P** *is* rank one and $\sigma_i = 0$ for $i \geq 2$. However, when signal perturbations are present, **P** has higher rank, but still has a few dominant singular values. Therefore, to get maximum compression we strive to approximate **P** by a matrix having a low rank while having a small MSE – in fact we will approximate it with a rank-one matrix. The effect of the time alignment is to concentrate the energy of the pulse matrix into the first singular value, producing a matrix that is closer to a rank one matrix. The effect of the noise on the singular values is uniformly spread across all the singular values – this is in fact a known result that is exploited in many applications of the SVD to signal processing problems. Thus, when we truncate the SVD to *k* terms as in (5) we are throwing away all the noise that exists in the thrown-away singular values, and – if we've done our job right – we have thrown away very little of the signal because it is mostly concentrated in the singular values that we keep. The effect of this is to ***increase*** the SNR of the reconstructed signal; thus, not only do we compress the signal but we get an improvement in SNR rather than a degradation due to compression! This simultaneous compression and noise reduction will be demonstrated in the simulations.

To extract a prototype pulse we consider the case where we truncate the SVD to a single term ($k = 1$) to get $\mathbf{P}_1$; we'll demonstrate later that the accuracy achieved with $k = 1$ is excellent (see also Figure 3). To specify $\mathbf{P}_1$ we need the $p \times 1$ vector $\mathbf{u}_1$ (i.e., the 1st *left* singular vector), the $n \times 1$ vector $\mathbf{v}_1$ (i.e., the 1st *right* singular vector), and the scalar σ_1 (i.e., the 1st singular value). Note that $\mathbf{v}_1^H$ is the same length as a pulse (*n* samples). Thus, we can interpret vector $\mathbf{v}_1^H$ as a single prototype pulse that has been extracted from the original pulse train, which is a nice viewpoint given that the *radar's* receiver would process *its* received pulse train using a pulse template as a matched filter; this leads to what we call a semi-coherent approach that uses the extracted prototype pulse as a matched filter at each platform.[15] Alternatively, we can view the reconstructed $\mathbf{P}_1$ as forming an approximation to the gated and aligned pulse train, which together with the alignment and gating side info can be used to create an approximation of the original pulse train. This latter viewpoint – what we call the coherent method – will be explored here.

In the coherent method we seek a reconstructed pulse *train* that would minimize the MSE between it and the original pulse train before it was compressed. For a given compression ratio (e.g., for a given number of terms retained in (5)) this clearly is the pulse train formed from the de-aligned rows of the approximating pulse matrix $\mathbf{P}_k$, due to the SVD minimizing the MSE. In addition to the prototype pulse we need the values of the left-singular vector $\mathbf{u}_1$, the fractional time alignments, and the number of samples between adjacent pulses that were removed by gating. The left-singular vector $\mathbf{u}_1$ is used to reassemble the truncated SVD form of the pulse matrix (up to the scaling factor of σ_1 – see below), after which the time alignment information is used to undo the time alignment and, finally, zeros are inserted in place of the gating-removed signal samples. Thus, for the cost of a modest amount of side information it is possible to reconstruct a MSE-minimizing pulse train suitable for coherent cross-correlation.

In particular, the approximating matrix $\mathbf{P}_1$ is formed from

$$\mathbf{P}_1 = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^H, \tag{6}$$

from which it is clear that each row in $\mathbf{P}_1$ is a complex-valued scalar multiple of $\mathbf{v}_1^T$, where the complex scalar for the i^{th} row is the i^{th} element in $\mathbf{u}_1$ times σ_1; it is also clear that σ_1 does nothing more than amplitude scale the entire reconstructed pulse train and can therefore be omitted. Thus, we can change (6) to

$$\widetilde{\mathbf{P}}_1 = \mathbf{u}_1 \mathbf{v}_1^H, \tag{7}$$

from which we see that $\mathbf{u}_1$ holds the reconstruction magnitudes and phases. Finally, the rows of $\widetilde{\mathbf{P}}_1$ have to be time shifted to undo the alignment processing time shifts, after which the results are assembled into a pulse train (with zeros inserted

between pulses to undo the effect of pulse gating) that is cross-correlated with the pulse train received locally at the Cx platform. Thus, the information that is needed to reconstruct the signal is:

1. The $n\times1$ right singular vector (RSV) $\mathbf{v}_1$ (i.e., the prototype pulse)
2. The $p\times1$ left singular vector (LSV) $\mathbf{u}_1$ (i.e., the reconstruction magnitudes and phases)
3. The p-1 time shifts ($\Delta_1, \Delta_2, \Delta_3, \ldots, \Delta_{p-1}$)
4. The p-1 numbers of inserted zeros ($N_1, N_2, \ldots N_{p-1}$)

Using this data at the Cx platform the reconstructed pulse train can be formed and then cross-correlated with the signal data received locally at the Cx platform to estimate the TDOA/FDOA.

2. Prototype Pulse Coding

We start here by focusing on coding the side information of $N_1, N_2, \ldots N_{p-1}$. The following characterization was done to show that it requires only a small number of bits to code the number of inserted zeros. First it should be recognized that it is likely that not every *transmitted* pulse in the pulse train will be intercepted and detected; thus, the spacing between pulses to be processed is unlikely to be constant or even nearly so. However, it is possible to specify a set number of samples that would be a greatest common divisor of the spacing – call it G; therefore, each sample spacing between sequential pulses in the detected pulse train will be integer multiples of G. It is likely that G could require something on the order of 16 bits to code. Denote by k_i the integer multiple of G that would then specify the i^{th} pulse spacing after gating. Based on an analysis of the k_i obtained for the field-collected radar data, we determined that "brute force" binary coding of the k_i could be done using 10 bits per k_i; however, using the measured radar data, a calculation of the information theoretic entropy of the k_i shows that it should be possible to code the k_i using Huffman or arithmetic coding having on average 4 bits per k_i. Alternatively, if we denote by Δk_i the difference between successive k_i it is seen by another calculation of information theoretic entropy from the data that it should be possible to code the Δk_i using about 2 bits per Δk_i on average by using Huffman or arithmetic coding. Thus, the number of bits needed to send the number of zeros inserted is on the order of $2p$+16 (where the 16 bits are for coding G), which is fairly small compared to the amount needed for the other data.

The rest of the data that must be coded consists of three parts: (i) the complex-valued prototype pulse (contained in the RSV), (ii) the magnitudes and phases of the reconstructed pulses (contained in the LSV), and (iii) the time-shifts of the pulses. To code the prototype pulse contained in the RSV we recognize that each of its samples is a complex number having magnitude and phase, both of which are changing from sample to sample. The cross-correlation processing will be much less sensitive to errors in the magnitude than in the phase, so we should ensure that the phase is coded with high accuracy whereas the magnitude can be coded with lower fidelity. We chose to code the phase of the prototype pulse using 8 bits/sample and to use a 1-bit differential PCM approach for the magnitudes of the prototype pulse. Thus, we use B_{RSV} = 8+1 = 9 bits to code each element of the RSV. It should be noted that this approach provides a fairly general approach that should work for virtually all cases; however, when the acquisition system identifies the radar as being a linear FM signal, the phase of the prototype should have fairly constant sample-to-sample phase differences, and then it may make more sense to use some form of differential coding there, too. To code the pulse magnitudes and phases contained in the LSV we again should allocate more bits to the phases than to the magnitudes. We use 4 bits per magnitude in the LSV and 8 bits per phase in the LSV. Thus, we use B_{LSV} = 8+4 = 12 bits to code each element of the LSV. Finally, each time shift is coded using B_{TS} = 8 bits.

How much compression can we get from this scheme? If no compression is used (other than gating) there are np complex samples to be sent and we use 8 bits/sample for the real part and 8 bits/sample for the imaginary part; thus, including the bits used to code the numbers of zeros to be inserted due to gating, the original signal is coded using

$$\text{Original Data} = 16pn + 2p + 16 \tag{8}$$

We first consider the case where a single pulse is put into each row of $p\times n$ **P**, but we will see later that it is usually better to put multiple pulses per row. We also only consider the case of keeping only 1 singular vector. Given the number of bits used to code the SVD compressed data, the total number of bits used for the compressed data is summarized in Table 2. Therefore, when using one pulse per row we get a compression ratio of

$$CR_{\text{subopt}} = \frac{16pn + 2p + 16}{22p + 9n + 6}, \tag{9}$$

which is labeled as suboptimum because we will see below that putting multiple pulses per row can improve the compression ratio.

As mentioned above, it is possible to improve this compression ratio by putting more than one pulse per row (after alignment) such that we now have an $r\times c$ matrix. This will require a few modifications, namely, we will need to normalize all the pulses so that even if we have significant pulse-to-pulse fading we will still be able to get a near rank one matrix. Thus, we won't have to code the magnitudes of the RSV, but we will now need p magnitude normalizers that can be coded using $B_{MN} = 2$ bits each. The $c\times 1$ RSV now only needs to have its phase coded, using $B_\phi = 8$ bits per element. This changes the results in Table 2 to those shown in Table 3.

Table 2: Total Compressed Data with One Pulse per Row

Quantity to Code	General Form of Coding	Specific Form of Coding
$n\times 1$ RSV	$(n \times B_{RSV})$	$(8+1)n = 9n$
$p\times 1$ LSV	$(p \times B_{LSV})$	$(8+4)p = 12p$
$(p\text{-}1)\times 1$ time shifts	$(p\text{-}1) \times B_{TS}$	$8(p-1) = 8p - 8$
$(p\text{-}1)\times 1$ # of inserted zeros	$(p\text{-}1) \times B_{NZ} + 16$	$2(p-1) + 16 = 2p + 14$
Compressed Data		$\mathbf{9n + 22p + 6}$

Table 3: Total Compressed Data with Multiple Pulses per Row

Quantity to Code	General Form of Coding	Specific Form of Coding
$(c\times 1)$ RSV Phases	$(c \times B_\phi)$	$8c$
$(r\times 1)$ LSV	$(r \times B_{LSV})$	$(8+4)r = 12r$
$(p\times 1)$ Magnitude Normalizers	$(p \times B_{MN})$	$2p$
$(p\text{-}1)\times 1$ time shifts	$(p\text{-}1) \times B_{TS}$	$8(p-1) = 8p - 8$
$(p\text{-}1)\times 1$ # of inserted zeros	$(p\text{-}1) \times B_{NZ} + 16$	$2(p-1) + 16 = 2p + 14$
Compressed Data		$\mathbf{8c + 12r + 12p + 6}$

The goal here is to find the optimal values of c and r. As a means of exploring this, for now assume that we can make any size pulse matrix for a given collection of pulses as long as the total number of elements equals the total number of samples np in the pulse train. Consider an $r\times c$ matrix with r and c chosen such that CR is maximized under the constraint that $rc = np$ (or equivalently that $r = np/c$). For this case the compression ratio becomes

$$\begin{aligned} CR &= \frac{16pn + 2p + 16}{8c + 12r + 12p + 6} \\ &= \frac{16pn + 2p + 16}{8c + \frac{12np}{c} + 12p + 6}, \end{aligned} \tag{10}$$

which should be maximized as a function of c for a given np. Thus, we must minimize the function

$$f(c) = 8c + \frac{12np}{c} + 12p + 6, \tag{11}$$

which is minimized when $c = \sqrt{3np/2}$ or equivalently when $r = \sqrt{2np/3}$; this is equivalent to making the pulse matrix such that $r = 2c/3$. From plots of CR vs. c, this peak is fairly broad so that hitting the exact value is not real crucial, so restricting r and c to be integers will not drastically reduce the CR from the theoretical maximum. Thus, we should put multiple pulses in a row in order to make the pulse matrix as close as possible to having two-thirds as many rows as columns. Using these results in (10), the optimal compression ratio is

$$CR_{\text{opt}} = \frac{16pn + 2p + 16}{8\sqrt{6np} + 12p + 6} \tag{12}$$

which is plotted in Figure 4 as a function of n and p. From this plot we see that for low to medium number of samples per pulse that the optimal compression ratio becomes effectively independent of the number of pulses as the number of pulses gets large. As both n and p increase, the compression ratio increases without bound; thus, we see that the compression ratio increases as the number of samples increases – that is, larger compression ratios are achieved when more compression is needed. Specific compression ratio results for typical practical scenarios are given in Table 4; the pulse width (PW), pulse repetition interval (PRI), and bandwidth (BW) values are for typical radars; the samples-per-pulse values are dictated by practical sampling theory; the ranges of number-of-pulses is dictated by the need to achieve sufficient TDOA/FDOA accuracy under expected conditions. It should be noted that these compression ratio results are much lower than some preliminary results[15] because those earlier results did not consider the impact of coding the side information; nonetheless, even including the effect of the side information, as we have here, the compression ratios achieved are still very good.

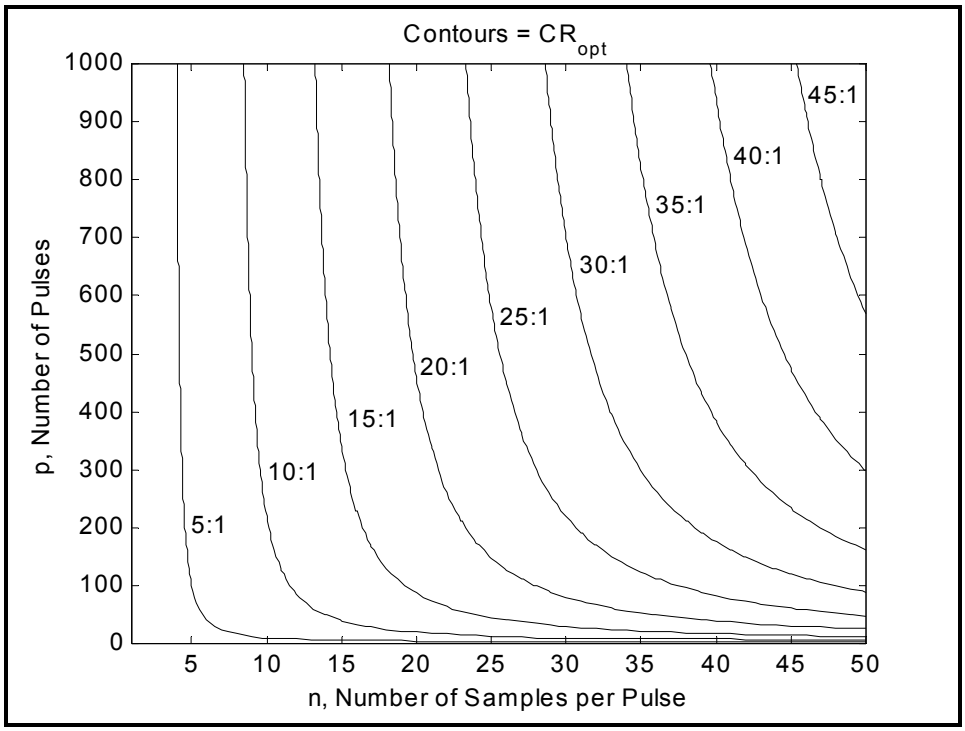

Figure 4: Contour plot showing optimal compression ratio as a function of the number of pulses and the number of samples per pulse.

Table 4: Compression Ratios for Typical Practical Scenarios

PW (μs)	**PRI** (μs)	**BW** (MHz)	**# of Pulses** p	**Samples/Pulse** n	CR_{subopt}	CR_{opt}
0.5	600	4.0	80 – 300	6	4.3 – 4.4	5.6 – 6.6
1.5	10	2.0	1,500 – 14,000	8	5.9 – 5.9	9.7 – 10.4
6.5	70	2.5	250 – 1,500	24	16.9 – 17.4	21.3 – 26.7
9.0	240	2.8	60 – 400	38	22.0 – 26.7	22.0 – 33.8

4. SIMULATION RESULTS

Monte Carlo simulations were performed to demonstrate the capability of the proposed method. The TDOA accuracy results are shown in Figure 5 and the FDOA accuracy results are shown in Figure 6, where it is seen that using the SVD

method actually improves the accuracy slightly despite the fact that it requires much less data transferal; at moderate SNR values the improvement is on the order of a 3 dB improvement in effective SNR. The case considered here had $p = 50$

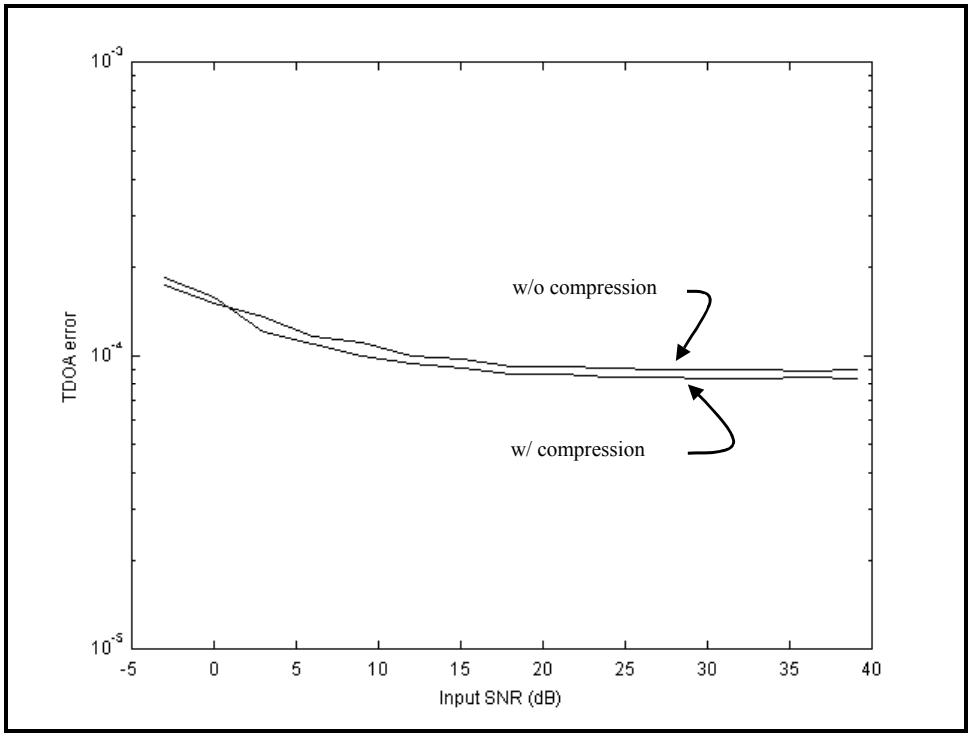

Figure 5: TDOA RMS Error Simulation Results with DNR = 20 dB; CR = 23:1.

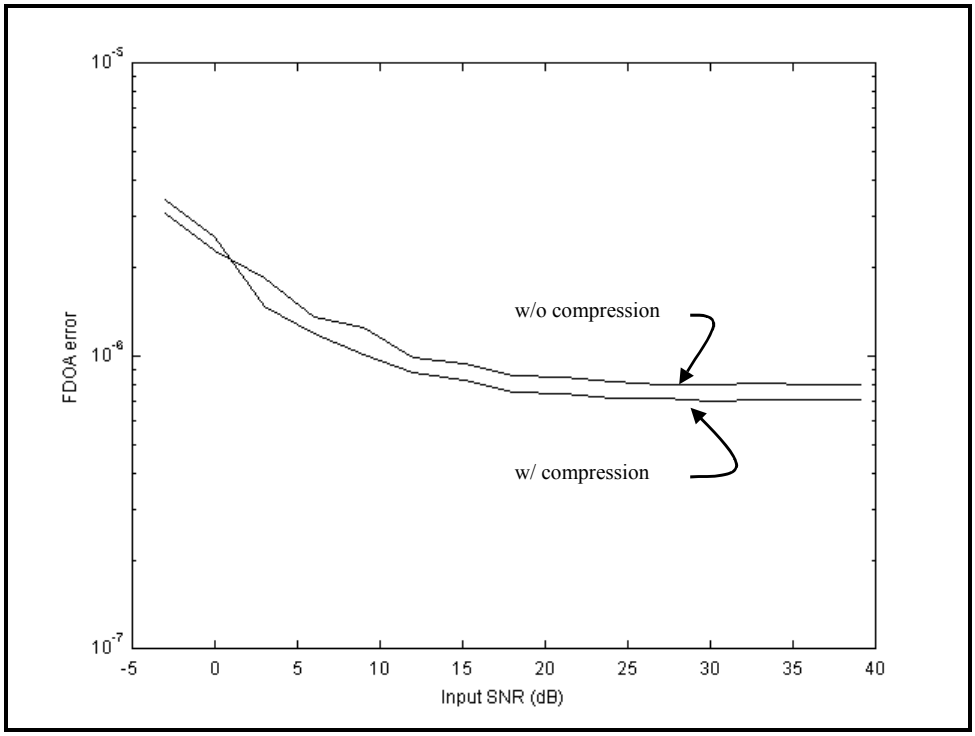

Figure 6: FDOA RMS Error Simulation Results with DNR = 20 dB; CR = 23:1.

pulses and $n = 43$ samples/pulse giving a compression ratio of 23:1. The SNR of the signal that was not compressed was set at $DNR = 20$ dB. It should be noted that these results are shown for *SNR* values lower than can be handled in practice due to the need to detect the pulses before gating and matrix formation; because we were interested in seeing how well the method worked at low SNR values we artificially used simulation-only information to gate the pulses at all SNR values.

5. CONCLUSIONS

From the above analyses we find that significant compression can be achieved at the expense of no accuracy degradation by following the following steps:

- Gate the pulses and put them into a pulse matrix with one pulse per row
- Align the pulses in the pulse matrix using fractional time shifts
- Reshape the pulse matrix to get as close as possible to having two-thirds as many rows as columns
- Compute the singular vectors (left and right) corresponding to the largest singular value
- Code the singular vectors and the side information

It is clear from the above results that the SVD is an effective means to exploit the pulse-to-pulse redundancy inherent in radar pulse trains. The key to its success is the ability to align the pulses and then exploit characteristics of the extracted data for efficient coding; in fact, without efficient coding of the number of gating-extracted samples, the compression ratio would decrease significantly. Although we used some real data to characterize how much data is needed to code this side information, more focus needs to be placed on this aspect of the method. Along these lines it will be worthwhile to take a close look at the gating process to find more efficient ways to code this side information.

The compression ratio result shows that this method gives a compression ratio that is equal to or better than other existing methods.[4 – 9] For example, noting that it is unlikely to ever have fewer than 5 samples per pulse we see from Figure 4 that we should expect a compression ratio higher than 5:1 regardless of the scenario. The best previously reported compression ratio is on the order of 7:1 for a particular linear FM radar pulse.[8] The results given here show that it is possible to get much larger compression ratios – up to about 30:1 or more for many practical scenarios.

Furthermore, an important characteristic of this method is that the amount of compression increases as the total number of samples increases – thus providing more compression when it is needed. This is extremely important for this application because the requirement usually imposes a desired time-to-location which requires more compression when the number of samples is large. However, it should be pointed out that if the increase in the total number of samples comes solely from increasing the number of pulses to be processes, then the increase in compression ratio is quite small.

By reshaping the aligned matrix to have two-thirds as many rows as columns it is possible to optimize the compression ratio for a given pair of n and p values. This reshaping is done by putting multiple pulses in a row, which results in the extracted prototype (the right singular vector) consisting, too, of multiple pulses; thus, we view this as extracting a prototype pulse trainlet rather than a prototype pulse. The amount of improvement due to this optimization can be significant and is therefore worth pursuing.

Finally, the SVD approach gives a very effective means of simultaneously compressing and reducing noise due to the fact that the noise is uniformly spread across the singular value spectrum while the signal is concentrated. The benefit of this is that there is no degradation in the TDOA/FDOA accuracies even when operating at high compression ratios.

6. REFERENCES

1. P. C. Chestnut, "Emitter location accuracy using TDOA and differential doppler ," *IEEE Trans. Aero. and Electronic Systems*, vol. AES-18, pp. 214-218, March 1982.
2. S. Stein, "Differential delay/doppler ML estimation with unknown signals," *IEEE Trans. Sig. Proc.*, vol. 41, pp. 2717 - 2719, August 1993.
3. S. Stein, "Algorithms for ambiguity function processing," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-29, pp. 588 - 599, June 1981.
4. D. J. Matthiesen and G. D. Miller, "Data transfer minimization for coherent passive location systems," Report No. ESD-TR-81-129, Air Force Project No. 4110, June 1981.
5. G. Desjardins, "TDOA/FDOA technique for locating a transmitter," US Patent #5,570,099 issued Oct. 29, 1996, held by Lockheed Martin.
6. M. L. Fowler, "Coarse quantization for data compression in coherent location systems," *IEEE Trans. Aero. and Electr. Systems*, vol. 36, no. 4, pp. 1269 – 1278, Oct. 2000.
7. M. L. Fowler, "Data compression for TDOA/DD-based location system," US Patent #5,991,454 issued Nov. 23, 1999, held by Lockheed Martin.

8. M. L. Fowler, "Data compression for emitter location systems," Conference on Information Sciences and Systems, Princeton University, March 15-17, 2000, pp. WA7b-14 – WA7b-19.
9. M. L. Fowler, "Decimation vs. quantization for data compression in TDOA systems," in *Mathematics and Applications of Data/Image Coding, Compression, and Encryption III*, Mark S. Schmalz, Editor, Proceedings of SPIE Vol. 4122, pp. 56 – 67, San Diego, CA, July 30 – August 4, 2000.
10. R. G. Wiley, *Electronic Intelligence: The Interception of Radar Signals*. Artech House, 1985.
11. R. G. Wiley, *Electronic Intelligence: The Analysis of Radar Signals*, 2nd Edition. Artech House, 1993.
12. T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Prentice Hall, 2000.
13. Z. Zhou, *Data Compression for Radar Signals: An SVD-Based Approach*, M.S. Thesis, State University of New York at Binghamton, May 2001.
14. T. I. Laakso, V. Valimaki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay," *IEEE Sig. Proc. Mag.*, vol. 13, no. 1, pp. 30 - 60, January 1996.
15. M. L. Fowler, Z. Zhou, and A. Shivaprasad, "Pulse Extraction for Radar Emitter Location," Conference on Information Sciences and Systems, Johns Hopkins University, March 21-23, 2001.